PENGEMBANGAN *BACKEND* SISTEM IOT BERBASIS *EXPRESS.JS* UNTUK MONITORING LINGKUNGAN DAN PENYEWAAN *ROVER* DI PERKEBUNAN KELAPA SAWIT

(Skripsi)

Oleh

I NENGAH MARCCEL JANARA BRATA CIPTA
NPM 2115031103

FAKULTAS TEKNIK
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG
2025

ABSTRAK

PENGEMBANGAN *BACKEND* SISTEM IOT BERBASIS *EXPRESS.JS* UNTUK MONITORING LINGKUNGAN DAN PENYEWAAN *ROVER* DI PERKEBUNAN KELAPA SAWIT

Oleh

I NENGAH MARCCEL JANARA BRATA CIPTA

Perkebunan kelapa sawit menghadapi tantangan dalam melakukan pemantauan lingkungan secara efisien, terutama pada area yang sulit dijangkau. Penelitian ini bertujuan mengembangkan sistem *backend* untuk mendukung penyewaan dan pemantauan perangkat *rover* berbasis *Internet of Things* (IoT). Sistem dibangun menggunakan *framework Express.js* dengan arsitektur modular guna memisahkan logika bisnis, validasi, *middleware*, dan konfigurasi, sehingga meningkatkan keteraturan dan skalabilitas.

Metode pengembangan yang digunakan adalah *Test-Driven Development* (TDD), yang terdiri dari tiga fase: *Red, Green*, dan *Refactoring*. Pada fase *Red*, sebanyak 138 kode pengujian ditulis menggunakan pustaka *Jest* sebelum implementasi. Fitur utama mencakup penyewaan berbasis langganan (6, 12, 24, dan 36 bulan), perhitungan biaya otomatis dengan diskon progresif hingga 20%, serta pengelolaan sensor, pengiriman, pemasangan, perpanjangan, dan pengembalian perangkat. Komunikasi dua arah antara perangkat IoT dan server menggunakan protokol MQTT, mendukung transmisi data suhu, kelembapan, dan intensitas cahaya secara waktu nyata, sekaligus memungkinkan pengendalian perangkat sesuai batas operasional harian.

Sistem diuji melalui 122 *unit test* pada modul *validator*, 171 *test case* integrasi, dan 149 *test case* di *Postman* dengan total 702 skenario, yang seluruhnya berhasil dengan cakupan 100%. Uji kerentanan dengan OWASP ZAP terhadap 47 plugin tidak menemukan kerentanan serius, hanya satu peringatan informasional. Sistem *backend* juga berhasil di-*deploy* ke VPS melalui *pipeline* CI/CD otomatis menggunakan *GitHub Actions* dan dijalankan secara stabil dengan PM2. Hasil penelitian menunjukkan bahwa sistem backend yang dikembangkan berfungsi dengan baik, aman dari kerentanan tingkat tinggi, serta mampu mendukung proses penyewaan dan pemantauan lingkungan berbasis *rover* pada perkebunan kelapa sawit.

Kata kunci: IoT, *rover*, *backend*, *Express.js*, MQTT, sewa berlangganan, TDD

ABSTRACT

DEVELOPMENT OF AN EXPRESS.JS-BASED IOT SYSTEM BACKEND FOR ENVIRONMENTAL MONITORING AND ROVER RENTAL IN OIL PALM PLANTATIONS

By

I NENGAH MARCCEL JANARA BRATA CIPTA

Oil palm plantations face challenges in achieving efficient environmental monitoring, particularly in areas that are difficult to access. This study develops a backend system to support the rental and monitoring of Internet of Things (IoT)-based rover devices. The system was implemented using the Express.js framework with a modular architecture to separate business logic, validation, middleware, and configuration, thereby enhancing maintainability and scalability.

The development process applied the Test-Driven Development (TDD) method, which consists of three phases: Red, Green, and Refactoring. During the Red phase, 138 test codes were written with the Jest library prior to implementation. The system's key features include subscription-based rental options (6, 12, 24, and 36 months), automated cost calculation with progressive discounts of up to 20%, and comprehensive device management covering sensors, delivery, installation, renewal, and return. Two-way communication between IoT devices and the server employs the MQTT protocol, enabling real-time transmission of temperature, humidity, and light intensity data, as well as device control based on daily operational limits.

The system was validated through 122 unit tests on the validator module, 171 integration test cases, and 149 Postman's test cases, with a total of 702 scenarios, all achieving 100% coverage. Vulnerability testing with OWASP ZAP on 47 plugins revealed no high-severity issues, with only one informational alert. The backend system was successfully deployed on a VPS through an automated CI/CD pipeline using GitHub Actions and runs reliably with PM2. The results demonstrate that the backend system performs effectively, with only minimal informational vulnerability risks, in supporting the leasing process and rover-based environmental monitoring for the palm oil plantations.

Keywords: IoT, rover, backend, Express.js, MQTT, subscription-based rental, TDD

PENGEMBANGAN *BACKEND* SISTEM IOT BERBASIS *EXPRESS.JS* UNTUK MONITORING LINGKUNGAN DAN PENYEWAAN *ROVER* DI PERKEBUNAN KELAPA SAWIT

Oleh

I NENGAH MARCCEL JANARA BRATA CIPTA

Skripsi

Sebagai Salah Satu Syarat untuk Mencapai Gelar
SARJANA TEKNIK

Pada

Jurusan Teknik Elektro
Fakultas Teknik Universitas Lampung

FAKULTAS TEKNIK
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG
2025

Judul Skripsi	:	**PENGEMBANGAN *BACKEND* SISTEM IOT BERBASIS *EXPRESS.JS* UNTUK MONITORING LINGKUNGAN DAN PENYEWAAN *ROVER* DI PERKEBUNAN KELAPA SAWIT**
Nama Mahasiswa	:	*I Nengah Marccel Janara Brata Cipta*
Nomor Pokok Mahasiswa	:	2115031103
Program Studi	:	Teknik Elektro
Fakultas	:	Teknik

MENYETUJUI

1. Komisi Pembimbing

Ing. Ardian Ulvan, S.T., M.Sc., Ph.D.
NIP. 197311281999031005

Aryanto, S.T., M.T.
NIP. 199006212019031011

2. Mengetahui

Ketua Jurusan
Teknik Elektro

Herlinawati, S.T., M.T.
NIP. 197103141999032001

Ketua Program Studi
Teknik Elektro

Sumadi, S.T., M.T.
NIP. 197311042000031001

MENGESAHKAN

1. Tim Penguji

Ketua : **Ing. Ardian Ulvan, S.T., M.Sc., Ph.D.** .....

Sekretaris : **Aryanto, S.T., M.T.** .....

Penguji : **Misfa Susanto, S.T., M.Sc., Ph.D.** .....

2. Dekan Fakultas Teknik

Dr. Eng. Ir. Helmy Fitriawan, S.T., M.Sc.
NIP. 197509282001121002

Tanggal Lulus Ujian Skripsi: **5 Agustus 2025**

SURAT PERNYATAAN

Yang bertanda tangan di bawah ini:

Nama	:	I Nengah Marccel Janara Brata Cipta
NPM	:	2115031103
Program Studi	:	Teknik Elektro
Jurusan/Fakultas	:	Teknik Elektro/Teknik
Alamat	:	Jl. Bali Indah, Desa Rejo Binangun, Kecamatan Raman Utara, Kabupaten Lampung Timur, Provinsi Lampung, 34371.

Dengan ini saya menyatakan bahwa dalam skripsi ini yang berjudul "Pengembangan *Backend* Sistem IoT Berbasis *Express.js* untuk Monitoring Lingkungan Dan Penyewaan *Rover* di Perkebunan Kelapa Sawit" tidak terdapat karya yang pernah dilakukan oleh orang lain dan sepanjang pengetahuan saya tidak terdapat atas diterbitkannya oleh orang lain, kecuali secara tertulis diacu dalam naskah ini sebagaimana yang disebutkan dalam daftar pustaka. Selain itu, saya menyatakan pula bahwa skripsi ini dibuat oleh saya sendiri.

Apabila pernyataan saya tidak benar, maka saya bersedia dikenai sanksi akademik sesuai dengan hukum yang berlaku.

Bandar Lampung, 19 September 2025

I Nengah Marccel Janara Brata Cipta
NPM. 2115031103

RIWAYAT HIDUP

Penulis lahir di Rejo Binangun pada tanggal 21 Januari 2003 sebagai anak kedua dari dua bersaudara dari pasangan Bapak (Alm.) Ketut Subrata dan Ibu Susiati. Penulis memulai pendidikan di SDN 2 Rejo Binangun pada tahun 2009 hingga 2015, SMPN 1 Raman Utara pada tahun 2015 hingga 2018 dan SMKN 1 Raman Utara pada tahun 2018 hingga 2021. Penulis menjadi mahasiswa Jurusan Teknik Elektro, Universitas Lampung melalui jalur SBMPTN (Seleksi Bersama Masuk Perguruan Tinggi Negeri). Selama menjadi mahasiswa, penulis aktif di Laboratorium Telekomunikasi sebagai asisten selama tahun 2023 s.d 2024. Pada semester 5, saya memilih mengambil konsentrasi Telekomunikasi dan Teknologi Informasi. Pencapaian saya yaitu berhasil Lolos Pendanaan Lomba Riset Sawit Tingkat Mahasiswa tahun 2024 sebagai ketua peneliti, membawa KOPMA Unila mendapatkan pendanaan pada kegiatan PPK Ormawa Kemendikbud Ristek tahun 2024, mengikuti kegiatan Kampus Merdeka Penelitian/Riset tahun 2023, memiliki satu paten sederhana yang terdaftar di DJKI dan menerbitkan sebuah artikel di jurnal nasional terakreditasi SINTA 2 pada tahun 2024. Selain itu, saya memiliki pengalaman magang selama satu semester di PT. Presentologic tahun 2024. Pencapaian ini menunjukkan dedikasi saya terhadap pengembangan diri, keterlibatan dalam bidang teknologi, serta keaktifan dalam berbagai kegiatan di lingkungan kampus. Saya berkomitmen untuk terus berperan dalam peningkatan kualitas dan pengembangan inovasi di dunia teknologi melalui perjalanan akademik yang saya tempuh.

PERSEMBAHAN

Om Awighnam Astu Namo Siddham

Atas Izin Ida Sang Hyang Widhi Wasa yang Maha Kuasa

KUPERSEMBAHKAN KARYA INI UNTUK

Ibu Tercinta

Ibu Susiati

Dan Kakakku Tersayang:

Wayan Neva Zullviana

Yang senantiasa menjadi sumber motivasi dan inspirasi, serta tak pernah lelah memberikan dukungan dan doanya:

Keluarga Besar Mbah Rian, Dosen, Almamater, TELTI 21 dan Mahasiswa Lab. Telti

Terima kasih untuk semua dukungan dan doa selama ini sehingga dapat menyelesaikan hasil karya ini

MOTTO

“Banyak kegagalan hidup dialami orang-orang yang tidak menyadari betapa dekatnya mereka dengan kesuksesan ketika mereka menyerah.”

(Thomas A. Edison)

“Pertumbuhan intelektual harus dimulai saat lahir dan berhenti saat kematian.”

(Albert Einstein)

“Dia yang memiliki alasan untuk hidup dapat menanggung hampir semua hal.”

(Friedrich Nietzsche)

“Setelah mendaki bukit besar, seseorang hanya menemukan bahwa masih banyak bukit lagi yang harus didaki.”

(Nelson Mandela)

SANWACANA

Om Awighnam Astu Namo Siddham. Dengan penuh rasa syukur ke hadapan Ida Sang Hyang Widhi Wasa, atas anugerah-Nya berupa tuntunan, kekuatan, dan ketekunan, penulis akhirnya dapat menyelesaikan penyusunan skripsi/tugas akhir dengan judul "Pengembangan *Backend* Sistem IoT Berbasis *Express.js* untuk Monitoring Lingkungan Dan Penyewaan *Rover* di Perkebunan Kelapa Sawit". Dalam proses penyusunan skripsi ini, penulis memperoleh banyak dukungan, baik secara moril maupun materiil, dari berbagai pihak yang telah memberikan semangat, arahan, serta bantuan yang sangat berarti. Untuk itu, dengan segala kerendahan hati, penulis menyampaikan terima kasih dan penghargaan setinggi-tingginya kepada semua pihak yang telah membantu, khususnya kepada:

1. Ibu tercinta dan seluruh keluarga penulis yang tidak hentinya mendo'akan serta memberikan dorongan semangat dan materi;
2. Bapak Dr. Eng. Helmy Fitriawan, S.T., M.Sc., selaku Dekan Fakultas Teknik Universitas Lampung;
3. Ibu Herlinawati, S.T., M.T. selaku Ketua Jurusan Teknik Elektro Universitas Lampung;
4. Bapak Ing. Ardian Ulvan, S.T., M.Sc., Ph.D. selaku Dosen Pembimbing Utama yang telah memberikan ilmu, bimbingan, bantuan, dan motivasi kepada penulis di setiap kesempatan dengan baik dan ramah;
5. Bapak Aryanto, S.T., M.T. selaku Dosen Pembimbing Pendamping yang telah memberikan ilmu, bimbingan, bantuan, motivasi, dan pandangan kehidupan kepada penulis di setiap kesempatan dengan baik dan ramah;
6. Bapak Misfa Susanto, S.T., M.Sc., Ph.D. selaku Dosen Penguji yang telah memberikan kritik dan saran yang membangun kepada penulis;
7. Segenap dosen di Jurusan Teknik Elektro yang telah memberikan ilmu yang bermanfaat kepada penulis selama menempuh pendidikan perkuliahan;

8. Segenap staff di Jurusan Teknik Elektro yang telah membantu penulis baik dalam hal administrasi dan lain-lain;
9. Keluarga Besar Laboratorium Telekomunikasi dan Informasi yang memberikan banyak ilmu, masukan, dan saran kepada penulis;
10. Muhammad Rhamadani, Bagus Munawar, Fadli Ferdinand Jaya, dan Luis Fernando Simbolon yang telah bersama-sama memberikan gagasan dalam menyelesaikan skripsi ini;
11. Muhammad Salman Abdurohman selaku rekan dalam menyelesaikan skripsi;
12. Seluruh pihak yang telah membantu penulis dalam menyelesaikan skripsi ini.

Penulis menyadari bahwa dalam penyusunan laporan skripsi ini masih terdapat berbagai kekurangan. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang bersifat membangun guna menjadi pembelajaran dan perbaikan di masa yang akan datang. Atas segala bantuan, dukungan, dan kontribusi yang telah diberikan oleh berbagai pihak dengan tulus, penulis menyampaikan rasa terima kasih yang sebesar-besarnya. Semoga laporan ini dapat memberikan manfaat bagi para pembaca serta menjadi kontribusi yang positif dalam bidang keilmuan terkait.

Bandar Lampung, 19 September 2025

I Nengah Marccel Janara Brata Cipta
NPM. 2115031103

DAFTAR ISI

Halaman

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

I PENDAHULUAN

1.1 Latar Belakang

Perkebunan kelapa sawit merupakan salah satu sektor penting yang berkontribusi besar terhadap perekonomian nasional. Sebagai komoditas strategis, kelapa sawit menempatkan Indonesia pada posisi sebagai produsen minyak sawit terbesar secara global. Pada tahun 2023, produksi tanaman kelapa sawit di Indonesia mencapai 46.986,10 ribu ton, menjadikannya salah satu sumber devisa terbesar bagi negara [1]. Meskipun demikian, pengelolaan perkebunan kelapa sawit masih dihadapkan pada tantangan signifikan dalam aspek pengawasan serta pemantauan kondisi lingkungan. Pemantauan ini menjadi penting untuk menjaga produktivitas dan keberlanjutan lingkungan perkebunan [2].

Metode pemantauan tradisional yang mengandalkan tenaga kerja manual sering kali tidak efisien, terutama untuk area perkebunan yang luas dan sulit dijangkau. Pemantauan manual tidak hanya memakan waktu tetapi juga meningkatkan risiko kesalahan dalam pencatatan data, yang dapat memengaruhi keputusan operasional [3]. Di tengah tantangan ini, kebutuhan akan teknologi berbasis IoT (*Internet of Things*) semakin mendesak. IoT dapat memberikan solusi untuk memantau kondisi lingkungan secara otomatis dan *real-time* (waktu nyata), namun implementasinya sering terhambat oleh biaya pengadaan perangkat yang tinggi serta kebutuhan akan keahlian teknis dalam pengelolaan sistem [4].

Salah satu inovasi dalam pengelolaan perkebunan adalah penggunaan perangkat *rover*. Perangkat ini memiliki kemampuan untuk mengumpulkan data lingkungan dari berbagai titik di perkebunan secara efisien [5], [6]. Namun, banyak perusahaan atau petani kecil yang tidak memiliki sumber daya untuk membeli perangkat ini secara langsung. Oleh karena itu, model bisnis penyewaan perangkat *rover* muncul sebagai solusi praktis untuk memberikan akses kepada pihak-pihak

yang memerlukan teknologi ini tanpa harus menanggung biaya pembelian yang besar. Kendati demikian, layanan penyewaan ini juga menghadapi tantangan, seperti kurangnya sistem *backend* yang andal untuk mengelola data pengguna, jadwal penyewaan, dan pengoperasian perangkat [7]. Dengan demikian, pengembangan sistem backend yang terintegrasi dengan model penyewaan menjadi solusi yang strategis, karena dapat mengurangi beban biaya dan kebutuhan keahlian teknis dari sisi pengguna.

Permasalahan lain yang sering dihadapi adalah kurangnya fitur kontrol jarak jauh untuk perangkat. Sebagai contoh, pengguna sering kali tidak dapat menghidupkan atau mematikan perangkat secara *real-time* dari lokasi yang berbeda. Keterbatasan ini mengurangi fleksibilitas dalam pengoperasian perangkat, khususnya ketika kondisi lingkungan membutuhkan respons cepat [8].

Untuk menjawab tantangan tersebut, penelitian ini bertujuan mengembangkan sistem *backend* berbasis *Express.js* guna mendukung penyewaan perangkat *rover* serta fitur pemantauan dan kontrol perangkat secara *real-time*. *Express.js* dipilih karena fleksibilitas dan kemampuannya dalam pengembangan aplikasi *backend* berbasis *Node.js*. Sistem *backend* ini menjadi fondasi penting dalam menjawab keterbatasan biaya dan keahlian teknis yang dihadapi oleh banyak petani dan pelaku usaha kecil, dengan menghadirkan solusi penyewaan teknologi berbasis IoT yang efisien dan mudah diakses. Perancangan *frontend* tidak menjadi fokus utama dalam penelitian ini, karena ruang lingkup pengembangan dibatasi hanya pada sisi *backend*.

Melalui penelitian ini, diharapkan dapat dihasilkan solusi teknologi yang tidak hanya mengoptimalkan pengelolaan perangkat *rover,* tetapi juga memperluas akses pengguna terhadap teknologi tersebut melalui model penyewaan yang lebih terjangkau dan fleksibel. Sistem *backend* yang dikembangkan dirancang untuk menyederhanakan proses manajemen penyewaan, pengumpulan data lingkungan secara *real-time*, serta memungkinkan kontrol perangkat dari jarak jauh. Sistem ini menggunakan MQTT (*Message Queuing Telemetry Transport*) sebagai protokol komunikasi, yang dikenal ringan dan efisien dalam mendukung interaksi perangkat IoT. Sementara untuk menjaga keamanan akses pengguna, digunakan JWT (*JSON Web Token*) sebagai metode otentikasi.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijabarkan di atas, rumusan masalah pada penelitian ini adalah sebagai berikut.

1. Bagaimana merancang dan mengimplementasikan sistem *backend* yang andal untuk mendukung pengelolaan penyewaan dan pengoperasian perangkat *rover* pada perkebunan kelapa sawit?
2. Bagaimana membangun *backend* yang mendukung komunikasi dua arah dengan perangkat *rover* melalui protokol MQTT untuk menerima data sensor dan mengirim perintah kontrol secara *real-time*?
3. Bagaimana merancang sistem *backend* yang aman dan efisien untuk komunikasi *real-time* antara perangkat *rover*, *backend*, dan aplikasi pengguna menggunakan *Express.js*?

1.3 Batasan Masalah

Agar penelitian ini memiliki ruang lingkup yang jelas dan fokus, maka batasan masalah ditentukan sebagai berikut:

1. Penelitian hanya mencakup pengembangan *backend* sistem menggunakan *Express.js*. *Frontend* atau antarmuka pengguna tidak menjadi bagian dari penelitian ini.
2. Sistem komunikasi data antara *rover* dan *backend* menggunakan protokol MQTT. Penelitian ini tidak mencakup pengembangan perangkat *rover* secara langsung, tetapi hanya menangani pengolahan data yang diterima dari perangkat tersebut dan kendali aktivasi perangkat.
3. Data yang dikelola meliputi suhu udara, kelembapan udara, dan intensitas cahaya. Penelitian ini tidak membahas jenis sensor tambahan di luar tiga parameter tersebut.
4. Penelitian ini hanya menyimulasikan pengelolaan data penyewaan perangkat *rover*, termasuk pencatatan pembayaran secara manual oleh *admin* berdasarkan bukti pembayaran yang diterima dari penyewa. Penelitian tidak mengimplementasikan fitur pembayaran otomatis melalui *payment gateway*.

Sistem *backend* yang dikembangkan hanya menyediakan mekanisme untuk mencatat status pembayaran dan menyimulasikan pengelolaan penyewaan tanpa proses transaksi digital yang sepenuhnya otomatis.

5. Penelitian ini hanya menghitung biaya sewa berdasarkan durasi kontrak bulanan tanpa mempertimbangkan jam operasional aktual perangkat. Pentarifan menggunakan tarif harian tetap yang dikalikan total hari kontrak dan diberi diskon sesuai durasi.

1.4 Tujuan Penelitian

Berdasarkan latar belakang yang telah dijabarkan di atas, tujuan pada penelitian ini adalah sebagai berikut.

1. Mengembangkan sistem backend berbasis *Express.js* untuk mendukung pengelolaan penyewaan perangkat *rover* secara efisien dan aman.
2. Membangun *backend* yang mendukung komunikasi dua arah dengan perangkat *rover* melalui protokol MQTT untuk menerima data sensor dan mengirim perintah kontrol secara *real-time*.
3. Merancang API (*Application Programming Interface*) yang aman dan efisien guna mendukung komunikasi antara sistem *rover*, *backend*, dan aplikasi pengguna, sehingga memungkinkan pengoperasian perangkat secara jarak jauh.

1.5 Manfaat Penelitian

Berdasarkan latar belakang yang telah dijabarkan di atas, manfaat pada penelitian ini adalah sebagai berikut.

1. Mempermudah pengelolaan dan pemantauan kondisi lingkungan perkebunan kelapa sawit secara secara *real-time* dengan menggunakan perangkat *rover* dan teknologi *backend* yang andal.
2. Meningkatkan efisiensi kerja melalui fitur kontrol perangkat jarak jauh, dan pengelolaan data transaksi penyewaan secara terpusat.
3. Mempercepat akses informasi pengguna melalui sistem pemantauan yang terintegrasi secara *real-time* dan terpusat.

II TINJAUAN PUSTAKA

2.1 Uraian Landasan Teori

2.1.1 *Internet of Things* (IoT)

Internet of Things (IoT) mengacu pada kumpulan perangkat pintar yang terhubung dalam suatu jaringan untuk saling berkomunikasi dan berbagi data. Perangkat dalam IoT dapat berupa sensor, aktuator, atau alat elektronik lainnya yang mampu menghasilkan informasi tentang lingkungannya dan bertindak berdasarkan masukan dari jaringan. Perangkat ini dapat melakukan identifikasi, pemantauan, dan pengelolaan secara otomatis melalui koneksi internet atau jaringan lokal. Penerapan IoT telah menjangkau berbagai sektor, termasuk kesehatan, transportasi, dan terutama pertanian yang memanfaatkan teknologi ini untuk memantau dan mengelola kondisi lingkungan secara efisien [9].

IoT telah membawa transformasi signifikan dalam bidang pertanian, terutama dalam penerapan konsep pertanian presisi dan pertanian terlindung (*protected agriculture*). Teknologi ini mendukung akuisisi data secara *real-time* dari sensor, yang digunakan untuk memantau parameter lingkungan seperti suhu, kelembaban, dan intensitas cahaya. Data ini dapat digunakan untuk mengoptimalkan pertumbuhan tanaman dan mencegah kerugian akibat perubahan lingkungan. Selain itu, IoT juga memungkinkan deteksi dini masalah seperti hama tanaman melalui kamera yang terintegrasi dalam sistem [10]. Implementasi IoT dalam pertanian tidak hanya mendukung pengelolaan tanaman tetapi juga dapat digunakan untuk pemantauan dan pengelolaan sumber daya alam secara berkelanjutan, mengurangi penggunaan air, dan mengoptimalkan penggunaan pupuk serta pestisida [11].

2.1.2 Rover

Sistem hibrida *rover* memiliki kemampuan penjelajah terestrial, sehingga memungkinkan aplikasi yang lebih serbaguna dan efisien di berbagai bidang, termasuk pertanian. Sistem ini menggabungkan keunggulan rover yang dapat menjangkau area luas dan medan sulit dengan stabilitas serta presisi di permukaan tanah. Konsep kendaraan *rover* hibrida sangat relevan di bidang pertanian, di mana rover menyediakan operasi di permukaan tanah yang presisi seperti pembersihan gulma dan penyemprotan. Penelitian telah menunjukkan kegunaan desain hibrida dalam mengatasi tantangan pertanian. Kendaraan hibrida ringkas dikembangkan untuk mengatasi hambatan seperti deteksi dan pengelolaan gulma melalui mekanisme penyemprotan dan pencabutan, yang menunjukkan hasil yang menjanjikan dalam tugas-tugas pertanian [12].

Selain itu, desain *rover* hibrida untuk pemantauan lingkungan menggunakan teknologi canggih seperti *Ground Penetrating Radar* (GPR) dan pembelajaran mesin untuk analisis data. Sistem semacam itu telah digunakan dalam karakterisasi tanah serta akuisisi data lingkungan secara akurat. Sistem hibrida ini mengintegrasikan sasis yang dibuat khusus, mekanisme keselamatan, dan platform komputasi yang efisien, yang menyoroti kemampuan beradaptasi mereka terhadap berbagai medan dan kebutuhan pengumpulan data [13].

Dalam perkembangan terbaru, sistem rover pertanian otonom mengintegrasikan visi komputer, pembelajaran mendalam, sensor tanah, dan aktuator robotik dalam satu platform. Sistem ini mampu mendeteksi kematangan tanaman dengan YOLOv5, melakukan pemetikan buah menggunakan lengan robotik enam derajat kebebasan (DOF), serta menganalisis hara tanah dengan sensor NPK. Dirancang untuk beroperasi secara mandiri atau jarak jauh, sistem ini menawarkan solusi efisien bagi negara berkembang yang menghadapi keterbatasan tenaga kerja dan akses teknologi [14].

2.1.3 Sistem *Backend*

Menurut GeeksforGeeks, pengembangan backend mencakup pengembangan sisi server, integrasi aplikasi, dan aktivitas seperti pembuatan pustaka dan penyusunan API. Karena berfokus pada arsitektur aplikasi, pemilihan teknologi yang tepat menjadi krusial untuk memilih teknologi pengembangan *backend* yang tepat. Ini berfungsi sebagai proses yang mengirim dan menerima informasi untuk berkoordinasi dengan *frontend* dan menampilkan data [15]. Bagian *backend* berfungsi untuk mendukung kinerja *frontend* agar dapat beroperasi dengan optimal. Peran seorang pengembang *backend* sangat penting dalam pembangunan sistem atau aplikasi yang mengelola data yang terus-menerus mengalami perubahan. Terdapat beberapa kriteria atau ciri-ciri pada *backend* software yang baik.

1. Memiliki kinerja yang baik

Kinerja merupakan kriteria pertama yang sangat penting dalam sistem *backend*. *Backend* yang baik harus mampu merespons permintaan dengan cepat dan efisien, sehingga pengguna tidak merasakan keterlambatan dalam penggunaan aplikasi. Kecepatan ini mencakup waktu respons server yang rendah, pemrosesan data yang efisien, serta kemampuan untuk menangani banyak permintaan secara bersamaan tanpa menurunkan kualitas layanan [16].

2. Memiliki keamanan yang baik

Keamanan adalah aspek krusial dalam pengembangan *backend*, terutama untuk aplikasi yang berinteraksi dengan data sensitif. *Backend* yang aman harus mampu melindungi data dari ancaman eksternal dan internal, seperti serangan injeksi *Structured Query Language* (SQL), serangan *Distributed Denial of Service* (DDoS), atau kebocoran data. OWASP ZAP (*Open Web Application Security Project Zed Attack Proxy*) digunakan pada tahap uji keamanan sebagai alat sumber terbuka yang mampu mengidentifikasi dan mengatasi kerentanannya dalam aplikasi web, termasuk dalam mendeteksi celah-celah keamanan berdasarkan standar OWASP Top 10 [17].

2.1.4 Express.js

Express.js adalah kerangka kerja web yang dibangun di atas *Node.js* dan berperan penting dalam pengembangan *backend* aplikasi web. Sebagai bagian dari stack MERN (*MongoDB, Express.js, React, Node.js*), *Express.js* berfungsi sebagai lapisan *middleware* yang mengelola routing, pengolahan data, dan autentikasi pada server. Dalam aplikasi *Study Buddy*, *Express.js* bertanggung jawab untuk memastikan komunikasi yang efisien dan aman antara server dan klien, serta mengatur alur data yang masuk dan keluar [18].

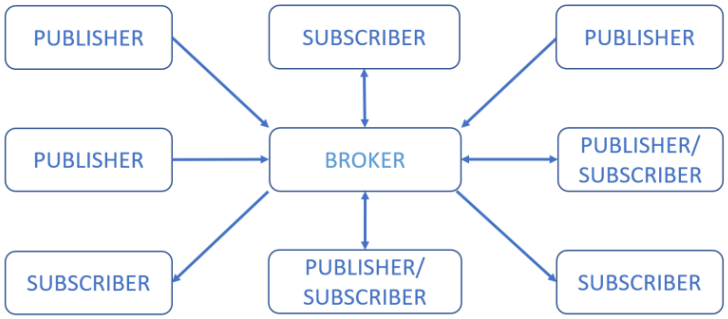
Express.js memiliki performa yang lebih ringan dibandingkan *framework* backend lainnya karena arsitektur *non-blocking* dan *event-driven* dari *Node.js*. Hal ini membuatnya cocok untuk membangun aplikasi berbasis RESTful API (Antarmuka Pemrograman Aplikasi berbasis REST) yang membutuhkan skalabilitas tinggi. Selain itu, *Express.js* mendukung berbagai teknik optimasi performa seperti *caching*, penggunaan *middleware* untuk validasi data, dan integrasi dengan database *NoSQL* seperti *MongoDB* yang mempercepat proses pengolahan data [19].

Dalam keamanan, *Express.js* juga menyediakan berbagai mekanisme perlindungan, termasuk *middleware* seperti *Helmet* untuk mengamankan *header* HTTP, *rate limiting* untuk membatasi jumlah permintaan dari satu sumber, serta dukungan terhadap autentikasi berbasis JWT (*JSON Web Token*). Dengan adanya fitur-fitur ini, *Express.js* menjadi pilihan utama dalam pengembangan *backend* modern yang membutuhkan kombinasi antara kinerja, fleksibilitas, dan keamanan. Selain itu, berdasarkan studi kasus, penerapan *Express.js* dalam aplikasi berbasis *microservices* juga semakin meningkat. Hal ini dikarenakan sifatnya yang ringan dan kompatibilitasnya dengan berbagai pustaka serta *framework* lain dalam ekosistem *JavaScript* [19].

2.1.5 MQTT (*Message Queuing Telemetry Transport*)

MQTT (*Message Queuing Telemetry Transport*) merupakan protokol komunikasi berbasis pesan yang bersifat ringan dan dikembangkan khusus untuk

perangkat dengan keterbatasan sumber daya dalam lingkungan *Internet of Things* (IoT). Protokol ini mengadopsi arsitektur *publisher/subscriber* dengan topologi bintang, yang memungkinkan komunikasi yang efisien antar perangkat dalam jaringan IoT. MQTT dapat digunakan dengan berbagai jenis perangkat, mulai dari sensor lingkungan, kendaraan, perangkat kesehatan, hingga perangkat industri. Protokol ini sangat cocok untuk digunakan pada perangkat dengan daya rendah, memori terbatas, dan harga terjangkau. Berdasarkan gambar 2.1, protokol MQTT memiliki tiga jenis peserta utama dalam sistem komunikasi, yaitu *broker, publisher*, dan *subscriber* [20].



Gambar 2. 1. Model *Publish/Subcribe* Protokol MQTT [20]

1. *Broker*, adalah pusat bintang dalam protokol MQTT dan bertanggung jawab atas pertukaran pesan antara peserta lain. Semua peserta lain terhubung dengannya dan hanya dengannya, sehingga ia bertanggung jawab atas otentikasi semua peserta dalam jaringan.
2. *Publisher*, adalah elemen yang mengirimkan data ke *broker* sehingga *broker* mengirimkan data ini ke satu atau lebih pelanggan yang berlangganan pada topik tersebut.
3. *Subcriber*, adalah elemen yang menerima data ke *broker*. Data yang mereka terima adalah data yang dikirim oleh *publisher*.

2.1.6 PostgreSQL

PostgreSQL merupakan sistem basis data objek-relasional *open-source* yang memiliki reputasi tinggi dalam hal keandalan, skalabilitas, serta kelengkapan fitur.

Antarmuka utamanya menggunakan SQL (*Structured Query Language*), yang menjadi standar pada sistem basis data relasional. *PostgreSQL* mendukung beberapa jenis indeks yaitu *BTree, Hash, Generalized Inverted Indexes* (GIN) dan *Generalized Search Tree* (GiST) yang disebut *R-tree-over-GiST*. Jenis indeks setelan awal adalah *BTree* yang dapat bekerja dengan semua tipe data dan dapat digunakan untuk persamaan dan kueri rentang secara efisien. Untuk struktur pohon yang seimbang secara umum dan kueri spasial berkecepatan tinggi, *PostgreSQL* menggunakan indeks GiST yang dapat digunakan untuk mengindeks tipe data geometris, serta pencarian teks lengkap [21].

2.1.7 Postman

Postman merupakan perangkat pengembangan API (*Application Programming Interface*) yang berfungsi untuk memudahkan pengembang dalam proses perancangan, pengujian, serta modifikasi API. Digunakan oleh lebih dari 5 juta pengembang setiap bulan, *Postman* menyederhanakan dan mempermudah proses pengembangan API. Alat ini mendukung berbagai fungsi yang dibutuhkan oleh pengembang API, termasuk kemampuan untuk membuat berbagai jenis permintaan HTTP (seperti *GET, POST, PUT, PATCH*), serta menyediakan fitur untuk menyimpan lingkungan kerja dan mengonversi API menjadi kode untuk berbagai bahasa pemrograman, seperti *JavaScript* dan *Python* [22]. Berikut adalah beberapa fitur utama yang menjadikan *Postman* alat yang sangat penting dalam pengembangan API:

1. *Postman* mendukung berbagai metode permintaan HTTP, termasuk *GET, POST, PUT, DELETE,* dan *PATCH*. Fleksibilitas ini memungkinkan pengembang untuk berinteraksi dengan API secara menyeluruh, mencakup berbagai operasi yang diperlukan dalam komunikasi API.
2. *Postman* memungkinkan pengembang untuk menangani berbagai format isi permintaan, seperti data formulir, data berkode URL, data mentah, dan data biner. Kemampuan ini memenuhi beragam kebutuhan pengembangan API, memungkinkan pengembang untuk bekerja dengan berbagai jenis data yang dipertukarkan dalam API.

3. *Postman* menyederhanakan proses autentikasi API dengan menyediakan dukungan untuk berbagai metode autentikasi, termasuk kunci API, *OAuth*, dan *Basic Auth*. Dengan dukungan ini, pengembang dapat dengan mudah mengamankan interaksi API, menjaga integritas dan keamanan data dalam aplikasi mereka.
4. *Postman* menyediakan fitur koleksi yang memungkinkan pengembang untuk mengatur dan mengelola permintaan API secara efisien. Koleksi ini turut memberikan kemudahan dalam proses berbagi serta kolaborasi antar anggota tim pengembangan. Selain itu, *Postman* memungkinkan pengujian otomatis menggunakan *JavaScript*, meningkatkan efisiensi dan efektivitas pengujian API.
5. *Postman* memudahkan pembuatan dokumentasi API secara langsung dari permintaan dan koleksi yang telah dibuat. Fitur dokumentasi ini menyediakan pendekatan terpusat yang efisien untuk mendokumentasikan API, yang bermanfaat bagi tim pengembangan internal dan pemangku kepentingan eksternal, memastikan bahwa dokumentasi tetap jelas dan mudah diakses.

2.1.8 Jest

Pengujian otomatis adalah proses pengujian perangkat lunak yang menggunakan teknologi, teknik, dan alat otomatisasi untuk menjalankan dan mengelola pengujian secara efisien dalam siklus pengembangan perangkat lunak. Proses ini mencakup aktivitas terstruktur yang dirancang untuk memastikan hasil yang diharapkan, dengan teknologi otomatisasi mencapai tingkat kecanggihan tertentu untuk memenuhi kebutuhan pengembangan perangkat lunak [23].

Salah satu alat yang populer untuk melakukan pengujian otomatis di ekosistem *JavaScript* adalah *Jest*. *Jest* merupakan kerangka kerja pengujian (*testing framework*) yang dikembangkan oleh Facebook [24]. Alat ini dirancang dengan fokus pada kesederhanaan dan dilengkapi dengan semua fitur yang diperlukan untuk memulai menulis pengujian secara langsung tanpa perlu konfigurasi

tambahan. Dengan *Jest*, pengembang dapat menulis pengujian secara efisien untuk berbagai kebutuhan, seperti pengujian unit, integrasi, hingga cuplikan.

2.1.9 *JSON Web Token* (JWT)

JSON Web Token (JWT) merupakan token keamanan berbasis JSON yang dirancang aman untuk digunakan dalam URL (*Uniform Resource Locator*) serta mampu memuat klaim yang dapat ditandatangani maupun dienkripsi. Spesifikasi JWT telah diadopsi secara luas karena mempermudah penyimpanan informasi keamanan di satu lokasi yang mudah dilindungi dan implementasinya sederhana dengan menggunakan alat yang tersedia secara umum. Salah satu aplikasi utamanya adalah untuk merepresentasikan informasi identitas digital, seperti token ID *OpenID Connect* dan token akses atau refresh dalam OAuth 2.0. Dalam praktik terbaiknya, JWT membutuhkan verifikasi algoritma yang ketat untuk memastikan kesesuaian header algoritma dengan operasi kriptografi, penggunaan algoritma yang sesuai dengan kebutuhan aplikasi, validasi seluruh operasi dan input kriptografi, serta memastikan kunci kriptografi memiliki tingkat entropi yang memadai. Selain itu, JWT harus memanfaatkan *encoding* UTF-8, menghindari kompresi pada input enkripsi, dan memvalidasi klaim seperti "*issuer*" untuk memastikan keamanan dan integritas data [25].

2.1.10 *Virtual Private Server (VPS)*

Virtual Private Server (VPS) adalah server pribadi yang seluruh sumber dayanya digunakan secara eksklusif oleh satu pengguna tanpa dipengaruhi pengguna lain. Teknologi ini mendukung eksekusi beberapa sistem operasi secara bersamaan pada satu mesin fisik melalui mekanisme virtualisasi perangkat keras. VPS menawarkan kendali penuh bagi pengguna untuk mengelola konfigurasi, termasuk akses penuh *(Full Root Access),* sistem operasi, init script, pengguna, pemrosesan, *filesystem*, hingga sumber daya server seperti CPU dan RAM. Dengan virtualisasi ini, satu server fisik dapat dibagi menjadi beberapa VPS yang berdiri sendiri, umumnya menggunakan sistem operasi Linux [26].

2.1.11 *GitHub dan Github Actions*

GitHub adalah platform kolaborasi yang merevolusi pengembangan perangkat lunak sumber terbuka dengan menyediakan alat seperti *issue reporting* dan *pull request*, yang terintegrasi dengan kontrol versi terdistribusi. Untuk mendukung otomatisasi alur kerja, *GitHub* menghadirkan *GitHub Actions*, sebuah fitur yang memungkinkan pengembang membuat *workflow* otomatis berdasarkan berbagai pemicu, seperti *commits* atau *pull requests. Workflow* ini ditulis dalam format YAML dan dapat mencakup berbagai tugas, seperti pengujian, integrasi berkelanjutan, hingga penerapan perangkat lunak. Contohnya, komunitas proyek Gammapy menggunakan *GitHub Actions* untuk menyambut kontributor baru melalui komentar otomatis. Penelitian menunjukkan bahwa *GitHub Actions* meningkatkan produktivitas pengembang dengan mengurangi tugas manual dan memengaruhi dinamika kontribusi, seperti jumlah komentar dan waktu penyelesaian *pull request*. Dengan meningkatnya popularitasnya, *GitHub Actions* kini memainkan peran penting dalam otomatisasi pengembangan perangkat lunak [27].

2.1.12 *Continuous Integration* (CI) dan *Continuous Deployment* (CD)

Continuous Integration (CI) merupakan praktik otomatisasi yang memastikan setiap perubahan kode dari pengembang langsung diintegrasikan ke repositori utama setelah dilakukan *commit*. Proses ini mencakup pengujian otomatis yang dilakukan secara cepat dan rutin untuk memastikan kode yang digabungkan tetap stabil dan bebas dari konflik. CI membantu tim pengembang menjaga kualitas aplikasi selama proses pengembangan dengan meminimalkan risiko integrasi kode yang bermasalah.

Sementara itu, *Continuous Deployment (CD)* adalah langkah lanjutan dari CI yang bertujuan untuk membangun dan merilis aplikasi secara otomatis setelah proses integrasi berhasil. Dengan CD, pembaruan aplikasi dapat dirilis dengan cepat dan tanpa intervensi manual, sehingga aplikasi selalu siap digunakan. Penggunaan CI/CD memberikan manfaat signifikan, seperti memperoleh *feedback*

lebih cepat, mendeteksi bug dengan segera, dan mempercepat proses rilis aplikasi. Hal ini menjadikan pengembangan perangkat lunak lebih efisien, andal, dan bebas dari bug [28].

2.1.13 Metode Pengembangan dengan *Test-Driven Development* (TDD)

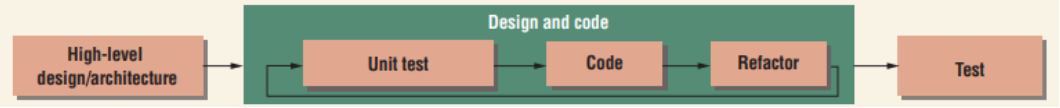
Menurut George & Williams, *Test-Driven Development* (TDD) adalah praktik pengembangan perangkat lunak yang menekankan penulisan *test case unit* sebelum implementasi kode, bagian dari pendekatan *Extreme Programming* (XP). Prinsip utamanya adalah bahwa pengembang hanya menulis kode untuk memenuhi kebutuhan berdasarkan tes yang telah ditentukan, sehingga menghasilkan desain yang lebih modular, kohesif, dan *loose coupling*, serta meminimalkan masalah seperti ketergantungan objek berlebih. Sebuah aturan penting dalam TDD adalah: "Jika Anda tidak dapat menulis tes untuk kode yang akan Anda buat, maka Anda seharusnya bahkan tidak berpikir untuk mulai menulis kode tersebut." TDD juga memungkinkan *continuous regression testing*, meningkatkan kualitas kode secara keseluruhan [29].

Fucci dkk. menyatakan bahwa *Test-Driven Development* (TDD) adalah teknik pengembangan perangkat lunak berbasis siklus yang sebaiknya diterapkan pada setiap implementasi fitur berskala kecil. Setiap siklus dalam TDD dimulai dengan menulis *unit test* untuk fitur yang akan dikembangkan, diikuti dengan implementasi kode yang diperlukan agar pengujian berhasil, dan diakhiri dengan proses *refactoring* untuk menghilangkan duplikasi, memperbaiki desain, atau mengganti perilaku sementara. Siklus ini juga memastikan bahwa semua *regression test* yang ada sebelumnya tetap berhasil. TDD menekankan iterasi mikro dalam setiap siklus pengembangan untuk menghasilkan kode yang lebih terstruktur dan teruji [30].

Berdasarkan definisi tersebut, *Test-Driven Development* (TDD) dapat disimpulkan sebagai pendekatan pengembangan perangkat lunak berbasis siklus yang mengharuskan penulisan *unit test* sebelum implementasi kode, sehingga menghasilkan desain yang modular, kohesif, serta mudah diuji, sekaligus mendukung *continuous regression testing*. Berikut adalah keuntungan *Test-Driven Development* (TDD) dalam pengembangan perangkat lunak [29]:

1. TDD membantu pemahaman kode dengan mendorong programmer menjelaskan kode menggunakan *test case* dan kode itu sendiri.
2. Memahami satu bagian kode dalam TDD memerlukan pengamatan terhadap test code dan kode utama, seperti prinsip "*measure twice, cut once*".
3. TDD menyediakan siklus pengujian dan kode yang sangat granular, memberikan umpan balik terus-menerus bagi programmer.
4. Kesalahan dapat diidentifikasi lebih cepat dengan TDD karena sumber masalah lebih mudah ditemukan saat kode baru ditambahkan.
5. Waktu tambahan untuk menulis dan menjalankan kasus pengujian pada TDD terbayar melalui efisiensi penghapusan cacat dan pengurangan waktu *debugging*.
6. TDD menghasilkan *unit test* otomatis yang menjadi aset berharga dalam pengembangan perangkat lunak.
7. *Unit test* otomatis dari TDD mendukung pengujian regresi untuk mendeteksi cacat baru dan memastikan konsistensi antar rilis perangkat lunak.
8. TDD mengurangi risiko pengenalan cacat baru selama *debugging* atau pemeliharaan dengan menjalankan *test cases* secara terus-menerus.

Tahapan metode pengembangan *Test-Driven Development* dapat dilihat pada gambar 2.2 [30].



Gambar 2. 2. Metode *Test-Driven Development [31]*

Berdasarkan gambar 2.2, alur proyek dimulai dengan mengidentifikasi arsitektur tingkat tinggi, tetapi desain tersebut tidak dilanjutkan hingga ke level yang sangat rinci pada tahap awal. Sebagai gantinya, pendekatan uji coba pertama mengutamakan penulisan *unit test* terlebih dahulu dan pengembangan unit kode dalam iterasi yang singkat dan cepat. Melalui proses ini, desain perangkat lunak dapat muncul dan berkembang secara bertahap, seiring dengan penambahan fitur

baru yang diuji dan dikodekan. Pendekatan ini memungkinkan desain perangkat lunak untuk berkembang seiring berjalannya waktu, dengan fokus pada pengujian setiap unit kecil secara terpisah [31].

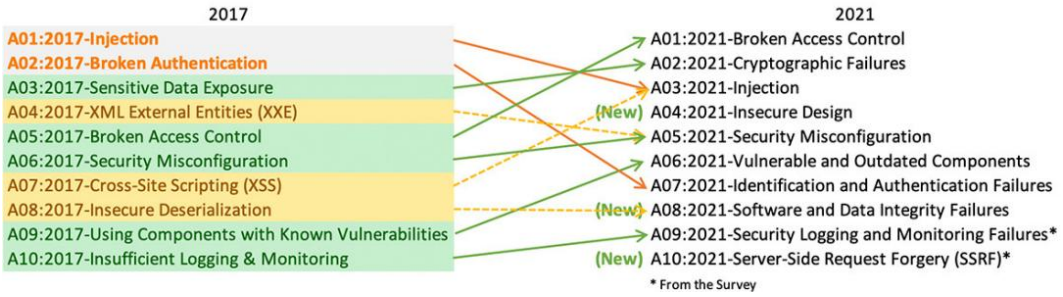
2.1.14 Kamus Data

Kamus data adalah elemen krusial dalam perancangan basis data yang berperan sebagai pusat metadata, mendeskripsikan setiap komponen data dalam sistem, meliputi nama atribut, tipe, ukuran, serta relasi antartabel. Kamus data tidak hanya membantu pengembang dalam memahami struktur basis data, tetapi juga menjadi referensi penting untuk validasi dan pengelolaan informasi selama siklus hidup basis data. Dalam konteks perancangan sistem informasi untuk bisnis konstruksi skala kecil dan menengah (SME), kamus data dirancang berdasarkan model konseptual ERD guna menjamin konsistensi dan interoperabilitas data. Setiap entitas dan atribut dijelaskan secara rinci untuk mendukung pengambilan keputusan dan pengembangan sistem informasi di masa depan [32].

2.1.15 OWASP ZAP

OWASP (*Open Web Application Security Project*) ZAP (*Zed Attack Proxy*) merupakan perangkat lunak keamanan aplikasi berbasis *open-source* yang dikembangkan untuk mendukung pengujian penetrasi serta mendeteksi kerentanan pada aplikasi web. ZAP memainkan peran penting dalam meningkatkan keamanan situs web dengan memindai dan mendeteksi potensi kerentanan, termasuk di antaranya masalah kontrol akses, kelemahan injeksi, dan desain yang tidak aman. Alat ini banyak digunakan di bidang keamanan siber karena antarmukanya yang mudah digunakan dan kemampuannya untuk memindai situs web secara otomatis untuk risiko keamanan yang umum. Alat OWASP ZAP tidak hanya menyediakan laporan kerentanan yang mendetail tetapi juga menyarankan tindakan pencegahan untuk mengatasi masalah yang ditemukan [17]. Sifatnya yang *open-source* dan aksesibilitas gratis membuatnya menjadi pilihan populer bagi para pengembang,

tim keamanan, dan organisasi yang ingin meningkatkan langkah-langkah keamanan web mereka.



Gambar 2.3. 10 Kerentanan Teratas OWASP Tahun 2021 [33]

Dalam sebuah studi kasus di situs web *crowdfunding syariah*, berbagai kerentanan keamanan diidentifikasi menggunakan OWASP ZAP, yang dikategorikan dalam 10 kerentanan teratas OWASP tahun 2021. Ini termasuk kontrol akses yang rusak, injeksi, desain tidak aman, kesalahan konfigurasi keamanan, komponen yang rentan, serta kegagalan menjaga integritas perangkat lunak dan data. Laporan tersebut menyoroti kerentanan kritis seperti Injeksi SQL, pengaturan *cookie* yang tidak aman, dan pustaka usang yang berpotensi membahayakan data sensitif. Kemampuan ZAP untuk menunjukkan dengan tepat masalah-masalah ini dan menawarkan strategi mitigasi sangat penting untuk meningkatkan keamanan aplikasi web secara keseluruhan. Berdasarkan gambar 2.3 OWASP Top 10 Kerentanan OWASP yang telah diperbarui semakin menekankan pentingnya mengatasi ancaman-ancaman ini untuk melindungi aplikasi web dari berbagai jenis serangan [33].

2.2 Penelitian Terdahulu

Penelitian ini merujuk pada sejumlah studi terdahulu yang relevan sebagai dasar pengembangan sistem.

a) Desain dan implementasi sistem pemantauan pencemaran lingkungan berbasis IoT: Studi kasus di Irak

Penelitian ini bertujuan untuk menerapkan teknologi *Internet of Things* (IoT) dalam pemantauan kualitas udara dan pencemaran lingkungan di Irak. Sistem yang

dikembangkan berfokus pada pengumpulan data pencemaran secara *real-time* guna mendukung pengambilan keputusan yang berkaitan dengan kesehatan masyarakat dan kebijakan lingkungan.

Penelitian ini menggunakan pendekatan berbasis sensor yang dipasang di berbagai titik lokasi tetap untuk mengukur parameter pencemaran udara seperti CO_2, NO_2, dan PM2.5. Sistem memanfaatkan perangkat IoT untuk mengirimkan data ke server, yang kemudian diproses, disimpan dalam *database MySQL*, dan ditampilkan melalui antarmuka web berbasis *dashboard*. Selain itu, algoritma tertentu diterapkan untuk mengevaluasi kualitas udara dan memberikan notifikasi bila kondisi membahayakan.

Meskipun pendekatan ini efektif untuk lokasi tetap, sistem tidak melibatkan perangkat bergerak seperti rover yang dapat menjangkau area dinamis atau sulit dijangkau. Kajian tersebut belum membahas pengelolaan data sensor secara *real-time* dari perangkat bergerak, serta belum mengintegrasikan pengembangan backend menggunakan *framework* seperti *Express.js* atau pendekatan *Test-Driven Development* (TDD) yang dapat meningkatkan reliabilitas sistem [34].

Oleh sebab itu, penelitian ini mengambil pendekatan berbeda pada aspek pemantauan lingkungan berbasis IoT, namun penelitian skripsi ini mengisi gap tersebut dengan mengembangkan sistem berbasis *rover* untuk pemantauan lingkungan secara dinamis di area perkebunan kelapa sawit. Sistem yang diusulkan juga menggunakan *Express.js* sebagai *backend* dan menerapkan TDD guna memastikan kualitas kode dan keandalan dalam pengelolaan data sensor secara *real-time*.

b) Rancang Bangun Sistem Penyewaan Baju dan Dekorasi Berbasis Web pada Nita *Wedding Organizer*

Studi ini mengembangkan sistem penyewaan berbasis web bagi Nita *Wedding Organizer*, yang memfasilitasi pelanggan dalam memilih dan memesan baju serta dekorasi untuk acara pernikahan secara *online*. Sistem ini mendukung penjadwalan dan pengelolaan inventaris untuk meningkatkan efisiensi dan kenyamanan pengguna.

Metode pengembangan yang digunakan dalam penelitian ini adalah model *Waterfall*, yang dilakukan secara bertahap mulai dari analisis kebutuhan hingga

pemeliharaan. Pengujian dilakukan setelah tahap implementasi untuk memastikan fungsionalitas sistem berjalan sesuai spesifikasi. Sistem ini mencakup fitur manajemen stok dan ketersediaan barang dalam penyewaan.

Sebaliknya, model *Waterfall* cenderung kurang fleksibel, khususnya apabila terdapat perubahan kebutuhan pengguna saat proses pengembangan sedang berlangsung. Selain itu, penelitian ini tidak mengadopsi pendekatan modern dalam rekayasa perangkat lunak seperti *Continuous Integration/Continuous Deployment (CI/CD), Test-Driven Development* (TDD), atau penerapan arsitektur berbasis *microservices* yang dapat meningkatkan skalabilitas dan keandalan sistem [35].

Relevansinya tetap terasa dalam pengembangan sistem penyewaan, meskipun penelitian ini memperluas konsep tersebut dengan menerapkannya pada penyewaan perangkat IoT berupa *rover* dalam konteks pertanian kelapa sawit. Sistem yang dikembangkan mengadopsi praktik TDD dan CI/CD untuk meningkatkan kualitas perangkat lunak, serta dirancang agar mampu menangani skenario penyewaan jangka panjang dan pengelolaan data perangkat IoT secara *real-time*.

c) Platform Web Jaringan Sensor Berbasis Teknologi WoT

Penelitian ini mengkaji pengembangan platform jaringan sensor berbasis *Web of Things* (WoT), yang merupakan perluasan dari konsep *Internet of Things* (IoT) dengan mengintegrasikan perangkat fisik dan sistem digital melalui protokol HTTP dan REST API. Tujuan utama dari penelitian ini adalah untuk membangun arsitektur platform WoT yang memungkinkan pengaksesan dan pengendalian perangkat pintar melalui antarmuka web secara efisien.

Studi tersebut mengimplementasikan protokol MQTT sebagai media komunikasi utama karena efisiensinya dalam menangani perangkat IoT dengan sumber daya terbatas melalui pendekatan *publish-subscribe*. Selain itu, platform dikembangkan menggunakan *Node.js* dan kerangka kerja *Express.js* untuk membangun RESTful API, serta memanfaatkan *Raspberry Pi* sebagai server web yang mempublikasikan data sensor dan aktuator ke internet secara *real-time*. Struktur URL yang terstandarisasi memudahkan integrasi perangkat dan komunikasi antar komponen sistem [36].

Namun, studi ini belum membahas secara rinci aspek keamanan seperti enkripsi, autentikasi, dan mitigasi risiko dalam komunikasi data. Selain itu, studi

ini kurang membahas ketahanan sistem terhadap tantangan operasional di lingkungan dengan konektivitas terbatas atau pada perangkat yang memiliki sumber daya sangat terbatas.

Adapun penelitian ini melengkapi kekurangan tersebut dengan tidak hanya menerapkan komunikasi dua arah berbasis MQTT, tetapi juga mengintegrasikan pengelolaan data sensor secara *real-time* dari perangkat bergerak seperti *rover* yang digunakan di lingkungan perkebunan kelapa sawit. Selain itu, sistem *backend* dikembangkan menggunakan *Express.js* dan memperhatikan aspek keamanan dan efisiensi data melalui *JSON Web Token* (JWT) untuk otentikasi, serta menerapkan arsitektur modular yang lebih adaptif terhadap keterbatasan koneksi dan perubahan topologi jaringan IoT di lapangan.

d) Sistem Penyewaan Lapangan Futsal Berbasis Web Kota Palembang Menggunakan *Express.js* dan *React.js*

Penelitian ini bertujuan merancang dan mengimplementasikan sistem penyewaan lapangan futsal berbasis web di Kota Palembang untuk meningkatkan efisiensi proses pemesanan. Sistem ini dirancang untuk menggantikan metode pemesanan konvensional yang mengharuskan calon penyewa datang langsung ke lokasi, yang tidak hanya memakan waktu tetapi juga tidak menjamin ketersediaan lapangan sesuai kebutuhan. Sistem ini memungkinkan pengguna mengakses informasi lapangan dan melakukan pemesanan secara *real-time*.

Dalam pengembangannya, penelitian ini menggunakan metode *prototype* yang mencakup tahapan identifikasi kebutuhan, perancangan, pembangunan, evaluasi, dan pengembangan akhir. *Express.js* digunakan sebagai *framework backend* untuk menangani manajemen data penyewaan, sedangkan *React.js* digunakan untuk merancang antarmuka pengguna yang interaktif dan responsif. Sistem ini memberikan kemudahan bagi pengelola dalam memperbarui informasi ketersediaan dan bagi pengguna dalam melakukan reservasi secara *online* [37].

Walaupun sistem ini fungsional untuk penyewaan lapangan, masih terdapat kekurangan yaitu tidak terdapat integrasi dengan sistem pembayaran otomatis yang dapat menyederhanakan proses transaksi, serta belum disediakan fitur pemantauan kondisi fisik lapangan. Selain itu, cakupan sistem terbatas pada fungsi penyewaan

berbasis waktu tanpa menyentuh aspek pemeliharaan fasilitas atau manajemen operasional lainnya.

Sementara itu, pendekatan teknologi dalam penelitian tersebut menjadi inspirasi bagi skripsi ini, terutama dalam penerapan *Express.js* sebagai fondasi *backend* sistem penyewaan. Akan tetapi, ruang lingkup dan kompleksitasnya diperluas untuk mendukung penyewaan perangkat *rover* yang dilengkapi dengan sensor untuk pemantauan lingkungan perkebunan kelapa sawit secara *real-time*. Sistem ini juga mendukung komunikasi dua arah menggunakan protokol MQTT, serta mengimplementasikan model bisnis *subscription-based rental* yang memungkinkan penyewaan perangkat dalam jangka waktu 6, 12, hingga 36 bulan. Dengan demikian, penelitian ini tidak hanya menangani proses penyewaan, tetapi juga mengintegrasikan monitoring lingkungan dan kontrol perangkat secara jarak jauh, yang belum dibahas dalam penelitian sebelumnya.

e) *Design of smart farming communication and web interface using MQTT and Node.js*

Kajian tersebut mengembangkan sistem komunikasi dan antarmuka web untuk *smart farming* dengan memanfaatkan protokol komunikasi MQTT dan platform *backend Node.js*. Tujuan utamanya adalah untuk meningkatkan efisiensi dalam pertanian presisi melalui pengumpulan dan pengelolaan data lingkungan secara *real-time*. Sistem ini dirancang agar data dari sensor dan kamera dapat dikirimkan ke server menggunakan MQTT, disimpan di basis data, dan ditampilkan melalui antarmuka web. Pengguna juga dapat melakukan kontrol perangkat secara jarak jauh melalui antarmuka tersebut.

Metodologi yang digunakan mencakup perancangan arsitektur komunikasi secara menyeluruh (*end-to-end*), pembuatan diagram *use case*, ERD (*Entity Relationship Diagram*), diagram alur pengguna, implementasi kode, dan pengujian sistem. Hasil pengujian menunjukkan bahwa sistem ini mampu menangani hingga 400 pengguna simultan dengan konsumsi RAM maksimum 389 MB, serta mampu menangani komunikasi dengan hingga 900 perangkat melalui protokol MQTT. Hasil studi menunjukkan bahwa kombinasi *Node.js* dan MQTT memberikan efisiensi tinggi dalam pemrosesan data dan kontrol perangkat pada sistem pertanian [38].

Kendati hasilnya menjanjikan, pendekatannya terbatas pada perangkat tetap dalam *smart farming* dan belum membahas integrasi sistem penyewaan maupun pengelolaan perangkat bergerak. Selain itu, belum dijelaskan pendekatan bisnis atau model operasional jangka panjang seperti penyewaan perangkat yang digunakan dalam sistem tersebut.

Sementara itu, penelitian ini menerapkan pendekatan teknologi serupa dalam konteks yang lebih luas, yaitu penggunaan *Node.js (Express.js)* untuk *backend* dan MQTT untuk komunikasi data sensor, namun diterapkan dalam konteks yang berbeda, yakni pengelolaan penyewaan dan pemantauan *rover* untuk mendukung efisiensi operasional di perkebunan kelapa sawit. Sistem yang dibangun dalam penelitian ini tidak hanya memungkinkan pemantauan lingkungan secara *real-time*, tetapi juga mendukung kontrol perangkat dari jarak jauh dan menawarkan skema *subscription-based rental*, memberikan solusi yang lebih komprehensif dibandingkan penelitian sebelumnya yang hanya berfokus pada sistem komunikasi dan antarmuka.

Dari tinjauan pustaka di atas, dapat disimpulkan bahwa meskipun teknologi seperti IoT, MQTT, dan *Express.js* telah banyak diterapkan secara terpisah, integrasi dalam sistem penyewaan berbasis perangkat bergerak dan kontrol dua arah masih jarang ditemukan. Oleh karena itu, penelitian ini mengisi gap tersebut dengan merancang sistem backend yang mendukung penyewaan perangkat *rover* sekaligus pemantauan data sensor secara *real-time* dan kontrol jarak jauh dalam perkebunan kelapa sawit.

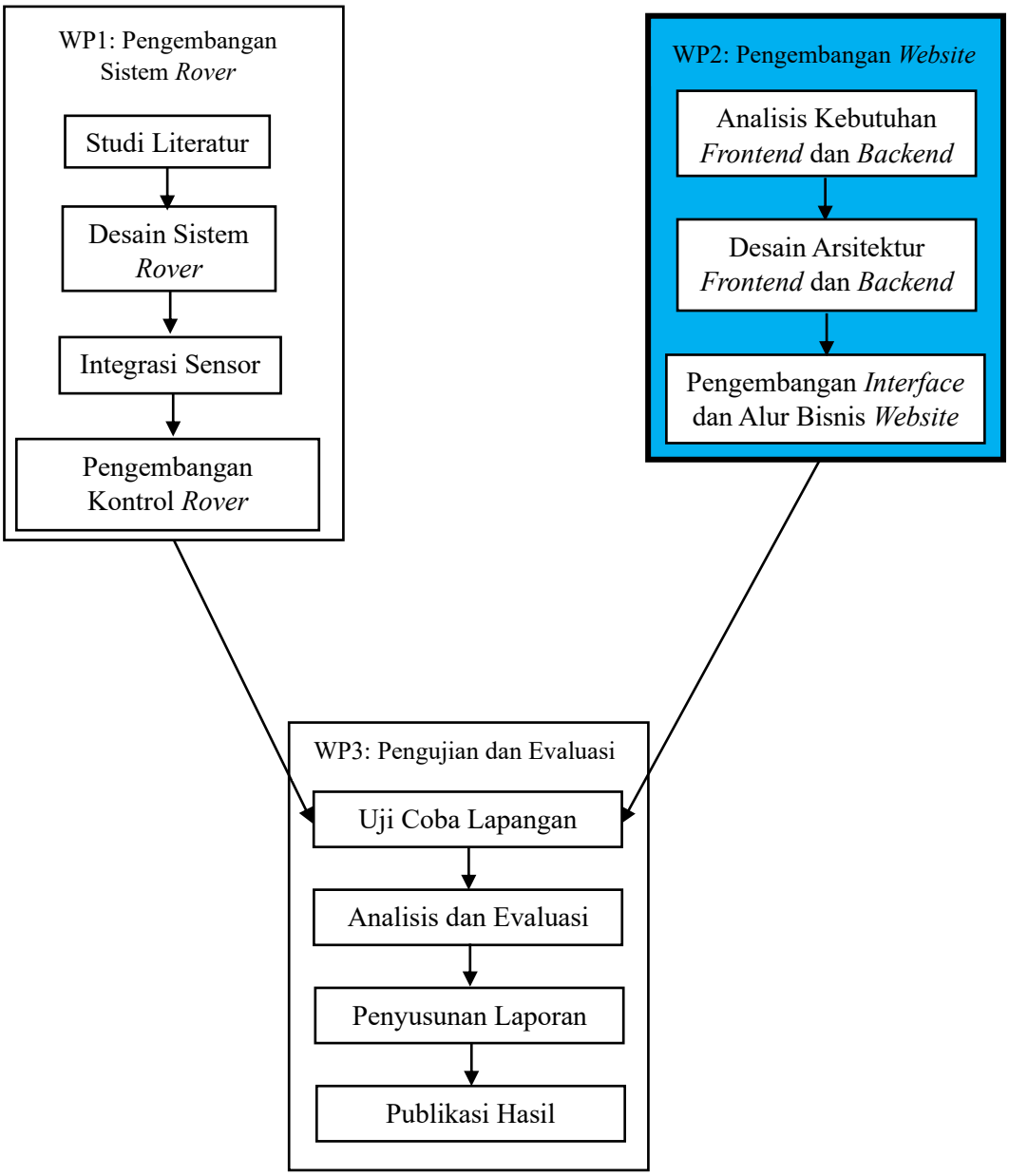
III METODE PENELITIAN

3.1 Waktu dan Tempat Penelitian

Penelitian ini dilaksanakan di Laboratorium Telekomunikasi Universitas Lampung pada periode 17 Desember 2024 hingga 31 Mei 2025. Kegiatan penelitian mencakup pengembangan *backend*, pengujian server, integrasi perangkat lunak, serta simulasi dan pengolahan data sensor lingkungan. Seluruh proses awal pengembangan dan pengujian dilakukan dalam lingkungan laboratorium yang terkontrol guna memastikan kelayakan dan stabilitas sistem. Setelah itu, dilakukan pengujian lapangan terhadap perangkat *rover* untuk mengamati performa sistem dalam kondisi nyata di luar laboratorium. Pendekatan bertahap ini memungkinkan peneliti untuk memastikan keandalan sistem sebelum diterapkan secara penuh di lingkungan perkebunan sesungguhnya.

3.2 Cakupan Penelitian

Penelitian ini merupakan bagian dari proyek besar bernama *rover*, yang dirancang untuk meningkatkan produktivitas perkebunan kelapa sawit dengan memanfaatkan teknologi *rover*. Dalam proyek ini, berbagai sistem terintegrasi seperti pengumpulan data sensor, pengendalian perangkat, dan penyimpanan data berbasis web digunakan untuk mendukung operasional. Fokus penelitian ini terletak pada pengembangan website, khususnya pada bagian *backend*.

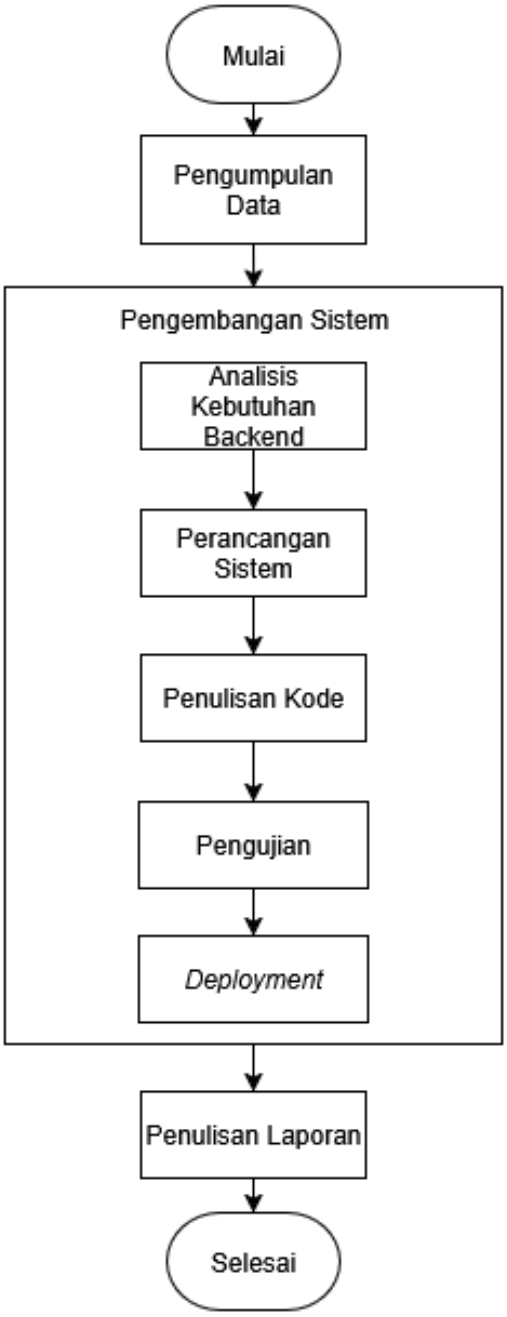


Gambar 3.1. Cakupan Penelitian

Berdasarkan Gambar 3.1, cakupan penelitian ini berada pada WP (*Work Package*) 2: Pengembangan Website, yang menjadi fokus utama penelitian ini. WP2 meliputi analisis kebutuhan, perancangan arsitektur, hingga pengembangan antarmuka dan alur bisnis website. Namun, ruang lingkup penelitian ini dibatasi pada pengembangan sisi backend, mencakup perancangan API, manajemen basis data, autentikasi, serta integrasi dengan sistem *rover* melalui komunikasi MQTT.

3.3 Tahapan Penelitian

Tahapan penelitian digambarkan pada Gambar 3.2, mencakup pengumpulan data, analisis kebutuhan, perancangan, pengujian sistem dan *deployment*. Setelah semua tahapan tersebut dilaksanakan, penulisan laporan akan dilakukan untuk mendokumentasikan seluruh proses dan hasil penelitian.



Gambar 3.2. Diagram Alir Tahapan Penelitian

3.3.1 Pengumpulan Data

Pengumpulan data menjadi tahap penting sebagai dasar pengembangan sistem *backend* yang sesuai kebutuhan pengguna. Dengan memahami kondisi lingkungan, tantangan yang dihadapi, dan harapan pengguna, pengembang dapat merancang solusi yang lebih baik dan fungsional. Data yang diperlukan akan diperoleh melalui beberapa metode sebagai berikut:

a. Studi Pustaka

Kajian pustaka mengacu pada literatur terkait pengembangan *backend*, komunikasi *Message Queuing Telemetry Transport* (MQTT), dan implementasi REST API. Selain itu, kajian pustaka juga dilakukan untuk memahami sistem penyewaan, termasuk mekanisme penyewaan berbasis teknologi, manajemen data transaksi penyewaan, dan sistem pembayaran digital. Literatur terkait studi kasus penyewaan perangkat IoT dan kendaraan berbasis sistem informasi digunakan untuk memberikan gambaran yang lebih mendalam tentang fitur-fitur yang perlu diimplementasikan. Studi literatur ini juga mencakup penelitian terdahulu yang dapat menjadi referensi dalam pengembangan sistem ini, baik dari segi teknologi yang digunakan maupun metodologi yang diterapkan.

b. Observasi

Observasi dilakukan secara langsung di lingkungan terbuka (lapangan) untuk memahami cara kerja perangkat *rover* dalam mengumpulkan data sensor seperti suhu udara, kelembapan udara, dan intensitas cahaya matahari. Fokus pengamatan adalah pengiriman data dari perangkat ke *backend* melalui MQTT secara *real-time*. Selain itu, diuji juga fitur kontrol jarak jauh, khususnya untuk menyalakan dan mematikan perangkat melalui perintah dari *dashboard*. Observasi turut mencakup interaksi pengguna dalam proses penyewaan, manajemen perangkat, dan penggunaan antarmuka sistem, guna memastikan kesesuaian antara rancangan sistem dengan kebutuhan operasional.

3.3.2 Pengembangan Sistem

Tahap ini menerapkan pendekatan *Test-Driven Development* (TDD) dalam pengembangan sistem. TDD merupakan metode yang menekankan penulisan pengujian sebelum pengembangan fitur, sehingga pengembang dapat memastikan bahwa setiap fungsi yang ditulis memenuhi spesifikasi yang diharapkan.

3.3.2.1 Analisis Kebutuhan *Backend*

Tahap analisis kebutuhan backend mencakup pengkajian kebutuhan sistem dan spesifikasi teknis yang menjadi dasar perancangan *backend*. Tujuannya adalah memahami kebutuhan pengguna, fungsionalitas sistem, dan persyaratan teknis maupun non-teknis. Analisis kebutuhan *backend* ini mencakup beberapa elemen, yaitu identifikasi data, identifikasi fungsional, identifikasi pengguna, analisis kebutuhan fungsional pengguna, serta analisis kebutuhan non-fungsional.

a. Identifikasi Data

Sistem backend ini mengelola berbagai jenis data, yaitu:

1. Data sensor
2. Data pengguna (*admin* dan *user*)
3. Data perangkat
4. Data penyewaan
5. Data pembayaran
6. Data notifikasi
7. Data laporan
8. Data pengiriman
9. Data pengembalian
10. Data alamat pengguna

b. Identifikasi Fungsional

Berdasarkan hasil observasi dan studi pustaka, maka dapat dilakukan identifikasi kebutuhan untuk sistem yang akan dibuat. Kebutuhan fungsional tersebut dapat didefinisikan sebagai berikut.

1. Admin dapat *login* sistem.

2. Admin dapat menambahkan perangkat baru
3. Admin dapat mendapatkan daftar perangkat
4. Admin dapat mendapatkan detail perangkat
5. Admin dapat menghapus perangkat
6. Admin dapat mengubah status kondisi perangkat
7. Admin dapat memperbarui topik sensor perangkat
8. Admin dapat memperbarui topik kontrol perangkat
9. Admin dapat mengirim notifikasi
10. Admin dapat mendaftarkan pengguna baru
11. Admin dapat menampilkan daftar pengguna
12. Admin dapat mendapatkan detail pengguna
13. Admin dapat menghapus akun pengguna
14. Admin dapat mengubah kata sandi pengguna
15. Admin dapat mengubah status/verifikasi rental
16. Admin dapat menghapus rental
17. Admin dapat mendapatkan semua daftar rental
18. Admin dapat melihat detail rental
19. Admin dapat mendapatkan daftar semua pembayaran
20. Admin dapat mendapatkan detail pembayaran
21. Admin dapat melakukan verifikasi pembayaran
22. Admin dapat menghapus pembayaran dengan *soft delete*
23. Admin dapat membuat laporan transaksi dalam rentang waktu
24. Admin dapat mendapatkan daftar laporan transaksi
25. Admin dapat mendapatkan detail laporan transaksi
26. Admin dapat mengunduh laporan dalam format PDF
27. Admin dapat menghapus laporan transaksi
28. Admin dapat mendapatkan detail pengiriman
29. Admin dapat mendapatkan daftar pengiriman
30. Admin dapat mengubah informasi pengiriman
31. Admin dapat memperbarui status pengiriman
32. Admin dapat konfirmasi tanggal pengiriman aktual
33. Admin dapat konfirmasi tanggal pengiriman diterima

34. Admin dapat mengunggah bukti pengiriman diterima
35. Admin dapat mendapatkan bukti pengiriman
36. Admin dapat mendapatkan detail pengembalian perangkat
37. Admin dapat memperbarui informasi pengembalian perangkat
38. Admin dapat mengambil semua data pengembalian perangkat
39. Admin dapat memperbarui status pengembalian perangkat
40. Admin dapat menambahkan catatan pengembalian perangkat
41. Admin dapat memperoleh detail perpanjangan masa sewa perangkat
42. Admin dapat memperoleh semua perpanjangan masa sewa perangkat
43. User dapat mendaftar akun
44. User dapat melakukan verifikasi akun
45. User dapat *login* sistem
46. User dapat menyewa perangkat
47. User dapat melihat daftar rental
48. User dapat melihat detail rental
49. User dapat membatalkan penyewaan tertunda
50. User dapat mendapatkan daftar perangkat
51. User dapat mendapatkan detail perangkat
52. User dapat melihat data sensor
53. User dapat mengunduh *dataset* sensor
54. User dapat mengontrol *rover*
55. User dapat menerima notifikasi penyewaan dan pembayaran
56. User dapat mendapatkan total penggunaan perangkat harian
57. User dapat mengajukan perpanjangan masa sewa perangkat
58. User dapat memperoleh detail perpanjangan masa sewa perangkat
59. User dapat memperoleh semua perpanjangan masa sewa perangkat
60. User dapat memperoleh semua daftar sensor yang tersedia
61. User dapat memperoleh hasil perhitungan biaya ke lokasi tujuan
62. User dapat mendapatkan detail pengiriman
63. User dapat memperoleh bukti pengiriman
64. User dapat memperoleh detail pengembalian perangkat
65. User dapat memperbarui alamat penjemputan maksimal dalam dua hari

66. User dapat menambahkan alamat pengiriman baru
67. User dapat mengambil semua alamat pengiriman
68. User dapat mengambil detail alamat pengirman
69. User dapat memperbarui alamat pengiriman
70. User dapat menetapkan alamat utama (*default*)
71. User dapat menghapus alamat

c. Identifikasi Pengguna

Berdasarkan hasil observasi dan studi pustaka, maka dapat dilakukan identifikasi pengguna untuk sistem yang akan dibuat. Identifikasi pengguna tersebut dapat dilihat pada tabel 3.1.

Tabel 3.1. Identifikasi Pengguna

No	Pengguna	Deskripsi
1	Pelanggan/u*ser*	Pengguna utama sistem yang terdiri dari individu atau organisasi yang menyewa perangkat *rover* untuk keperluan pemantauan lingkungan perkebunan.
2	*Admin*	Pengelola sistem yang bertanggung jawab atas operasional *backend*, infrastruktur, dan layanan pelanggan.

d. Analisis Kebutuhan Fungsional Pengguna

Berdasarkan hasil observasi dan studi pustaka, maka dapat dilakukan analisis kebutuhan fungsional untuk sistem yang akan dibuat. Analisis kebutuhan fungsional pengguna tersebut dapat dilihat pada tabel 3.2.

Tabel 3.2. Analisis Kebutuhan Fungsional

No	Pengguna	Fungsional	Deskripsi
1	*Admin*	Masuk ke dalam sistem	Proses autentikasi *admin* agar dapat mengakses sistem *backend*.
2	*Admin*	Menambahkan perangkat baru	*Admin* dapat menambahkan perangkat baru ke dalam sistem untuk digunakan dalam penyewaan.

No	Pengguna	Fungsional	Deskripsi
3	*Admin*	Mendapatkan daftar perangkat	Menampilkan daftar seluruh perangkat yang tersedia dalam sistem.
4	*Admin*	Mendapatkan detail perangkat	Menampilkan informasi detail dari perangkat tertentu.
5	*Admin*	Menghapus perangkat	Menghapus perangkat dari sistem dengan metode *soft delete*.
6	*Admin*	Mengubah status kondisi perangkat	*Admin* dapat memperbarui status perangkat, seperti aktif atau tidak aktif.
7	*Admin*	Memperbarui topik sensor perangkat	Memperbarui topik MQTT untuk komunikasi data sensor perangkat.
8	*Admin*	Memperbarui topik kontrol perangkat	Memperbarui topik MQTT yang digunakan untuk mengontrol perangkat dari sistem.
9	*Admin*	Mengirim notifikasi	Merupakan proses otomatis mengirimkan notifikasi *email* ke pengguna terkait proses penyewaan dan pembayaran
10	*Admin*	Mendaftarkan pengguna baru	Admin dapat membuat akun baru untuk pengguna dalam sistem.
11	*Admin*	Menampilkan daftar pengguna	Menampilkan daftar seluruh pengguna yang terdaftar dalam sistem.
12	*Admin*	Mendapatkan detail pengguna	Menampilkan informasi detail dari pengguna tertentu.
13	*Admin*	Menghapus akun pengguna	*Admin* dapat menonaktifkan atau menghapus akun pengguna
14	*Admin*	Mengubah kata sandi pengguna	*Admin* dapat memperbarui kata sandi pengguna jika diperlukan.
15	*Admin*	Mengubah status/verifikasi rental	*Admin* dapat mengubah status penyewaan, antara lain menyetujui, menolak permohonan dan penyelesaian rental.
16	*Admin*	Menghapus rental	Menghapus data penyewaan dari sistem dengan metode *soft delete.*
17	*Admin*	Mendapatkan semua daftar rental	Menampilkan daftar seluruh transaksi penyewaan yang pernah dilakukan oleh pengguna.

No	Pengguna	Fungsional	Deskripsi
18	*Admin*	Melihat detail rental	Menampilkan informasi detail dari transaksi penyewaan tertentu.
19	*Admin*	Mendapatkan daftar semua pembayaran	Menampilkan daftar seluruh transaksi pembayaran dalam sistem.
20	*Admin*	Mendapatkan detail pembayaran	Menampilkan informasi detail dari transaksi pembayaran tertentu.
21	*Admin*	Melakukan verifikasi pembayaran	Memeriksa dan mengonfirmasi pembayaran yang dilakukan oleh pengguna.
22	*Admin*	Menghapus pembayaran dengan *soft delete*	Menandai transaksi pembayaran tertentu sebagai dihapus tanpa benar-benar menghapusnya dari sistem.
23	*Admin*	Membuat laporan transaksi dalam rentang waktu	Menghasilkan laporan transaksi dalam periode tertentu berdasarkan data sistem.
24	*Admin*	Mendapatkan daftar laporan transaksi	Menampilkan daftar laporan transaksi yang telah dibuat.
25	*Admin*	Mendapatkan detail laporan transaksi	Menampilkan informasi detail dari laporan transaksi tertentu
26	*Admin*	Mengunduh laporan dalam format PDF	Memungkinkan admin untuk mengunduh laporan transaksi dalam format PDF untuk dokumentasi.
27	*Admin*	Menghapus laporan transaksi	Menghapus data laporan transaksi dari sistem.
28.	*Admin*	Mendapatkan detail pengiriman	Menampilkan detail pengiriman berdasarkan ID pengiriman.
29.	*Admin*	Mendapatkan daftar pengiriman	Menampilkan seluruh data pengiriman perangkat yang tercatat di sistem.
30.	*Admin*	Mengubah informasi pengiriman	Memperbarui informasi seperti ekspedisi, alamat tujuan, dan catatan pengiriman.
31.	*Admin*	Memperbarui status pengiriman	Mengubah status pengiriman (dikirim, dalam perjalanan, terkirim).

No	Pengguna	Fungsional	Deskripsi
32.	*Admin*	Mengonfirmasi tanggal pengiriman aktual	Mencatat tanggal aktual saat perangkat benar-benar dikirim.
33.	*Admin*	Mengonfirmasi tanggal pengiriman diterima	Mencatat tanggal saat perangkat diterima oleh penyewa.
34.	*Admin*	Mengunggah bukti pengiriman diterima	Mengunggah foto atau dokumen sebagai bukti penerimaan perangkat.
35.	*Admin*	Mendapatkan bukti pengiriman	Menampilkan file bukti pengiriman yang telah diunggah.
36.	*Admin*	Mendapatkan detail pengembalian perangkat	Menampilkan informasi detail terkait pengembalian perangkat.
37.	*Admin*	Memperbarui informasi pengembalian perangkat	Mengubah informasi pengembalian
38.	*Admin*	Mengambil semua data pengembalian perangkat	Menampilkan seluruh data pengembalian perangkat yang tercatat.
39.	*Admin*	Memperbarui status pengembalian perangkat	Mengubah status pengembalian (misal: dikembalikan, sedang diperiksa, selesai).
40.	*Admin*	Menambahkan catatan pengembalian perangkat	Admin dapat menambahkan catatan pengembalian perangkat
41	*Admin*	Memperoleh detail perpanjangan masa sewa perangkat	Melihat informasi lengkap dari pengajuan perpanjangan sewa tertentu.
42	*Admin*	Memperoleh semua perpanjangan masa sewa perangkat	Melihat seluruh riwayat pengajuan perpanjangan sewa.
43	*User*	Mendaftar akun	Proses pendaftaran akun baru oleh pengguna untuk dapat mengakses sistem.

No	Pengguna	Fungsional	Deskripsi
44	*User*	Melakukan verifikasi akun	Pengguna melakukan verifikasi akun melalui kode OTP yang dikirim ke email.
45	*User*	Masuk akun ke dalam sistem	Proses autentikasi pengguna agar dapat mengakses layanan dalam sistem.
46	*User*	Menyewa perangkat sensor	Proses penyewaan rover untuk pemantauan lingkungan, serta mendapatkan akses kontrol perangkat melalui MQTT.
47	*User*	Membatalkan penyewaan tertunda	Pengguna dapat membatalkan penyewaan yang masih dalam status menunggu konfirmasi.
48	*User*	Melihat daftar rental	Menampilkan daftar seluruh transaksi penyewaan yang pernah dilakukan oleh pengguna.
49	*User*	Melihat detail rental	Menampilkan informasi detail dari transaksi penyewaan tertentu.
50	*User*	Mendapatkan daftar perangkat	Menampilkan daftar perangkat yang tersedia untuk disewa oleh pengguna.
51	*User*	Mendapatkan detail perangkat	Menampilkan informasi detail dari perangkat tertentu yang tersedia dalam sistem.
52	*User*	Melihat data sensor	Memantau data lingkungan seperti suhu, kelembapan, dan intensitas cahaya secara *real-time* melalui *dashboard*.
53	*User*	Mengontrol rover	Mengaktifkan atau menonaktifkan perangkat melalui aplikasi, di mana perintah dikirim ke *backend* dan diteruskan ke perangkat via MQTT.
54	*User*	Mengunduh dataset sensor	Mengakses dan mengunduh data sensor dalam format *timeseries* untuk keperluan analisis lebih lanjut.
55	*User*	Menerima notifikasi penyewaan dan pembayaran	Menerima pemberitahuan terkait penyewaan dan pembayaran melalui email.

No	Pengguna	Fungsional	Deskripsi
56	User	Mendapatkan total penggunaan perangkat harian	Melihat total jam penggunaan perangkat untuk hari ini.
57	User	Mengajukan perpanjangan masa sewa perangkat	Mengirim permintaan untuk memperpanjang durasi sewa perangkat.
58	User	Memperoleh detail perpanjangan masa sewa perangkat	Melihat informasi lengkap dari pengajuan perpanjangan sewa tertentu.
59	User	Memperoleh semua perpanjangan masa sewa perangkat	Melihat seluruh riwayat pengajuan perpanjangan sewa.
60	User	Memperoleh semua daftar sensor yang tersedia	Melihat semua jenis sensor yang dapat ditambahkan pada perangkat.
61	User	Memperoleh hasil perhitungan biaya ke lokasi tujuan	Melihat estimasi ongkos kirim perangkat ke alamat tujuan.
62	User	Mendapatkan detail pengiriman	Melihat informasi detail pengiriman perangkat sewaan.
63	User	Mendapatkan bukti pengiriman	Melihat atau mengunduh bukti pengiriman perangkat (foto/dokumen).
64	User	Memperoleh detail pengembalian perangkat	Melihat informasi detail proses pengembalian perangkat.
65	User	Memperbarui alamat penjemputan	Mengubah alamat penjemputan perangkat maksimal H-2 sebelum penjemputan.
66	User	Menambahkan alamat pengiriman baru	Menyimpan alamat baru yang akan digunakan untuk pengiriman perangkat.
67	User	Mengambil semua alamat pengiriman	Menampilkan seluruh daftar alamat pengiriman milik pengguna.
68	User	Mengambil detail alamat pengiriman	Melihat informasi detail dari salah satu alamat pengiriman yang dimiliki.

No	Pengguna	Fungsional	Deskripsi
69	User	Memperbarui alamat pengiriman	Mengubah informasi alamat yang telah tersimpan.
70	User	Menetapkan alamat utama (default)	Menjadikan salah satu alamat sebagai alamat utama/*default* untuk pengiriman.
71	User	Menghapus alamat	Menghapus salah satu alamat dari daftar alamat yang tersimpan.

e. Analisis Kebutuhan Non Fungsional

Untuk mencapai kinerja optimal dan keamanan yang memadai, sistem ini memerlukan sejumlah kebutuhan non-fungsional yang mencakup aspek keamanan, perangkat keras, dan perangkat lunak. Berikut adalah rincian kebutuhan tersebut.

a. Kebutuhan Keamanan

1. Otentikasi dan Otorisasi

   Sistem memerlukan otentikasi yang andal menggunakan *JSON Web Token (JWT)* agar setiap pengguna mendapatkan akses yang sesuai dengan peran mereka, baik sebagai pengguna biasa (*user*) maupun *admin*. Memastikan hanya pengguna yang sah dapat mengakses aplikasi.

2. Enkripsi Data

   Data sensitif, termasuk kredensial pengguna dan komunikasi antara *backend* dan perangkat *rover*, harus dilindungi melalui enkripsi TLS/SSL. Enkripsi ini menjaga agar data tetap aman dan terhindar dari akses tidak sah selama proses transmisi.

3. Kontrol Akses MQTT

   Pengaturan kontrol akses pada *MQTT broker* diperlukan untuk memastikan bahwa hanya pengguna yang diotorisasi yang dapat mengakses dan berinteraksi dengan perangkat melalui topik tertentu.

4. Proteksi Terhadap Serangan

   Sistem harus melindungi integritas dan keamanannya dengan menerapkan perlindungan terhadap serangan umum, seperti SQL

Injection, XSS, dan CSRF. Langkah ini dilakukan untuk meminimalkan potensi kerentanan dan menjaga data tetap aman.

b. Kebutuhan Perangkat Keras

Kebutuhan perangkat keras dalam membangun sistem ini adalah sebagai berikut.

1) Laptop

System Manufacturer : Asus ROG GL503GE

System Type: 64-bit operating system, x64-based processor

Processor : Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz

Memory: 24.0 GB RAM

2) Virtual Private Server

OS: Ubuntu 24.04 LTS

CPU core: 2

Memori: 8 GB

Bandwith: 8 TB

Disk space: 100 GB

3) Perangkat *Rover* dengan Sensor IoT

Sistem mengandalkan *rover* yang dilengkapi dengan modul komunikasi MQTT dan perangkat keras untuk mendukung fungsi *remote (aktivasi),* serta dilengkapi sensor pengukur suhu, kelembapan dan intensitas cahaya. Sensor-sensor ini memungkinkan pemantauan lingkungan secara *real-time*, sementara fungsi *remote* memberikan kemampuan kontrol perangkat dari jarak jauh.

c. Kebutuhan Perangkat Lunak

1) Platform *Backend*

Sistem *backend* dikembangkan menggunakan Node.js dengan *framework Express.js*. Pilihan ini mendukung kebutuhan akan REST API yang responsif, skalabel, dan mampu menangani koneksi dengan perangkat *rover* secara efisien.

2) Basis Data

PostgreSQL dimanfaatkan sebagai sistem basis data guna menyimpan data sensor dengan cara yang terstruktur dan efisien.

Database ini mendukung pencarian data *timeseries* yang memudahkan akses historis dan analisis kondisi lingkungan dari data yang terkumpul.

3) *MQTT Broker*

Mosquitto atau *MQTT broker* lainnya diperlukan untuk mengelola komunikasi *real-time* antara perangkat sensor pada *rover* dan sistem *backend*. Protokol MQTT ini memungkinkan pengiriman dan penerimaan perintah secara langsung, sehingga memudahkan kontrol dan monitoring perangkat dari jarak jauh.

4) Sistem Pengelolaan Pesan dan Notifikasi

Untuk pemberitahuan kode OTP pendaftaran dan teknis penyewaan, sistem menggunakan layanan pengelola email yaitu *Nodemailer*, yang memungkinkan notifikasi dikirim kepada pengguna.

5) Keamanan dan Otentikasi

Library *jsonwebtoken* digunakan untuk memberikan keamanan utama pada fitur *login*. Selain itu, *jsonwebtoken* (JWT) mendukung otentikasi berbasis token, memastikan akses pengguna sesuai dengan peran masing-masing, serta keamanan interaksi dalam sistem.

6) Visual Studio Code

Visual Studio Code dimanfaatkan sebagai *Integrated Development Environment* (IDE) utama dalam proses pengembangan. *Visual Studio Code* menyediakan fitur lengkap untuk penulisan, *debugging*, serta pengelolaan proyek berbasis *Node.js* dan *Express.js*, mendukung produktivitas tim pengembang.

7) Pengujian API dan Otomasi

Jest digunakan untuk menguji fungsionalitas *backend* melalui metode *unit testing* dan *integration testing*, sementara Postman digunakan untuk menguji *endpoint* API secara manual maupun otomatis menggunakan fitur Postman Collection Runner. Selain itu, *GitHub Actions* dimanfaatkan untuk mengotomatiskan

proses *Continuous Integration* (CI) dan *Continuous Deployment* (CD), sehingga dapat menjamin kestabilan aplikasi sebelum dirilis ke lingkungan produksi.

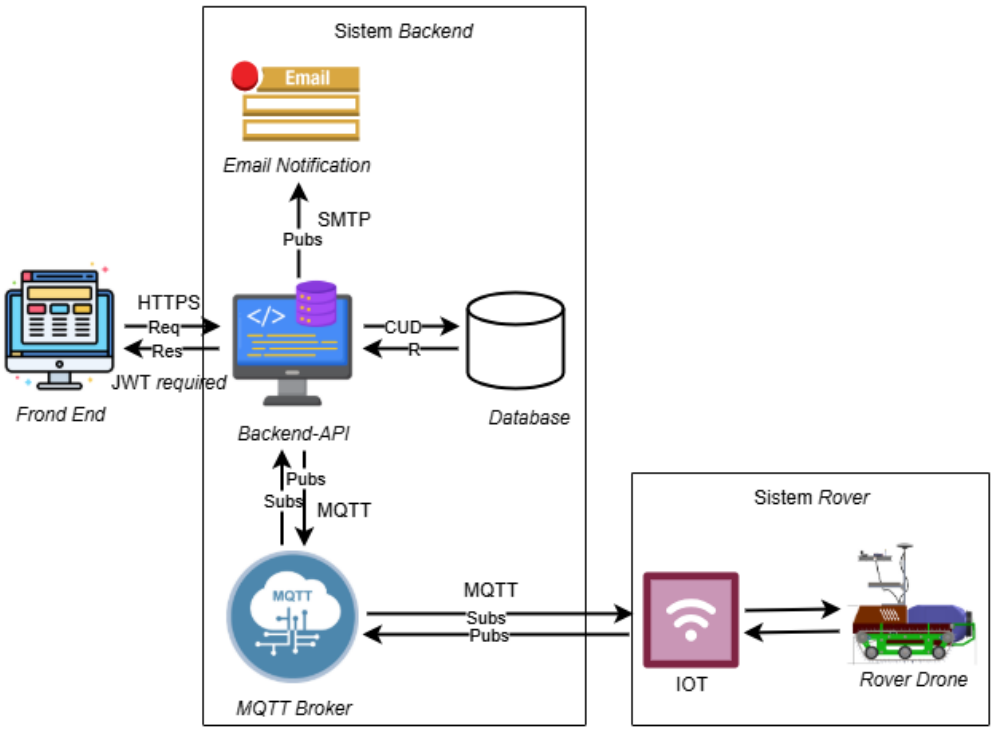
8) Keamanan API dan Pengujian Keamanan

OWASP ZAP digunakan untuk melakukan *security testing* terhadap API *backend* guna mendeteksi potensi celah keamanan dan kerentanan dalam sistem.

3.3.2.2 Perancangan Sistem

1. Arsitektur Sistem

Sistem yang dirancang memiliki tiga komponen utama, yaitu *frontend*, sistem *backend*, dan sistem *rover*. Ketiga subsistem berkomunikasi melalui protokol yang aman dan efisien guna menjamin performa serta keamanan sistem. Gambar 3.3 berikut memberikan gambaran arsitektur sistem secara keseluruhan.



Gambar 3.3. Arsitektur Sistem

Berdasarkan gambar 3.3, dalam perancangan sistem, dilakukan penguraian lebih lanjut terhadap:

a. Komponen Sistem

1) Frontend (antarmuka pengguna)

Frontend adalah antarmuka pengguna yang digunakan oleh pelanggan dan admin untuk berinteraksi dengan sistem. Fungsi utama *frontend* meliputi:

a) Menyediakan akses ke fitur sistem, seperti pengelolaan data penyewaan, laporan, dan notifikasi.

b) Mengirimkan permintaan HTTP ke *Backend-API* menggunakan protokol HTTPS untuk keamanan data.

2) Sistem *Backend* (sisi server aplikasi)

Sistem *backend* berfungsi sebagai pusat pengolahan data dan logika aplikasi. Komponen utama dalam backend meliputi:

a) Komponen *backend-API* menyediakan *endpoint* untuk menerima dan memproses permintaan dari *frontend* serta mengelola data yang dikirim oleh perangkat IoT melalui *MQTT broker* (perantara komunikasi dalam protokol *Message Queuing Telemetry Transport*).

b) Komponen *database* bertugas menyimpan data pengguna, perangkat, penyewaan, laporan transaksi, data sensor, dan notifikasi.

c) Komponen sistem notifikasi mengirimkan notifikasi push email kepada pengguna, terutama *One Time Password (OTP)* saat proses pendaftaran dan notifikasi teknis penyewaan.

d) Komponen MQTT broker berfungsi sebagai perantara komunikasi *real-time* antara perangkat IoT, *rover*, dan *backend*.

3) Sistem *Rover*

Sistem *rover* terdiri atas perangkat IoT yang terintegrasi dengan sensor untuk mengukur suhu, kelembapan, dan intensitas cahaya. Fungsi utama sistem ini meliputi:

a) Mengumpulkan data lingkungan secara *real-time* dan mengirimkannya ke *MQTT Broker*.

b) Mendukung kontrol perangkat secara *remote*, seperti fungsi *aktivasi rover.*

b. Alur Komunikasi Antar Komponen

Alur komunikasi dalam sistem dirancang untuk memastikan pengolahan data berjalan secara sinkron dan efisien:

1) Komunikasi antara *Frontend* dan *Backend*

a) Pengguna mengakses fitur sistem melalui *frontend*. Permintaan data atau tindakan dikirimkan ke *backend-API* menggunakan protokol HTTPS.

b) *Backend-API* memproses permintaan tersebut dengan mengambil atau menyimpan data dari *database*.

2) Komunikasi antara *Backend* dan *Database*

Backend-API berinteraksi langsung dengan *database* untuk membaca dan menyimpan data, seperti informasi penyewaan, data sensor, laporan, dan notifikasi.

3) Komunikasi antara *Backend* dan Sistem Notifikasi

Backend mengirimkan notifikasi melalui *email* kepada pengguna ketika proses pendaftaran, informasi perubahan kata sandi dan teknis terkait penyewaan.

4) Komunikasi antara *Backend* dan MQTT *broker*

Backend-API menggunakan MQTT *Broker* untuk menerima data dari perangkat IoT atau *rover*. MQTT broker memastikan komunikasi berlangsung secara *real-time*.

5) Komunikasi antara sistem *rover* dan MQTT *broker*

Sistem *rover* mengirimkan data sensor lingkungan ke MQTT *broker*, yang kemudian diteruskan ke *backend* untuk penyimpanan dan analisis.

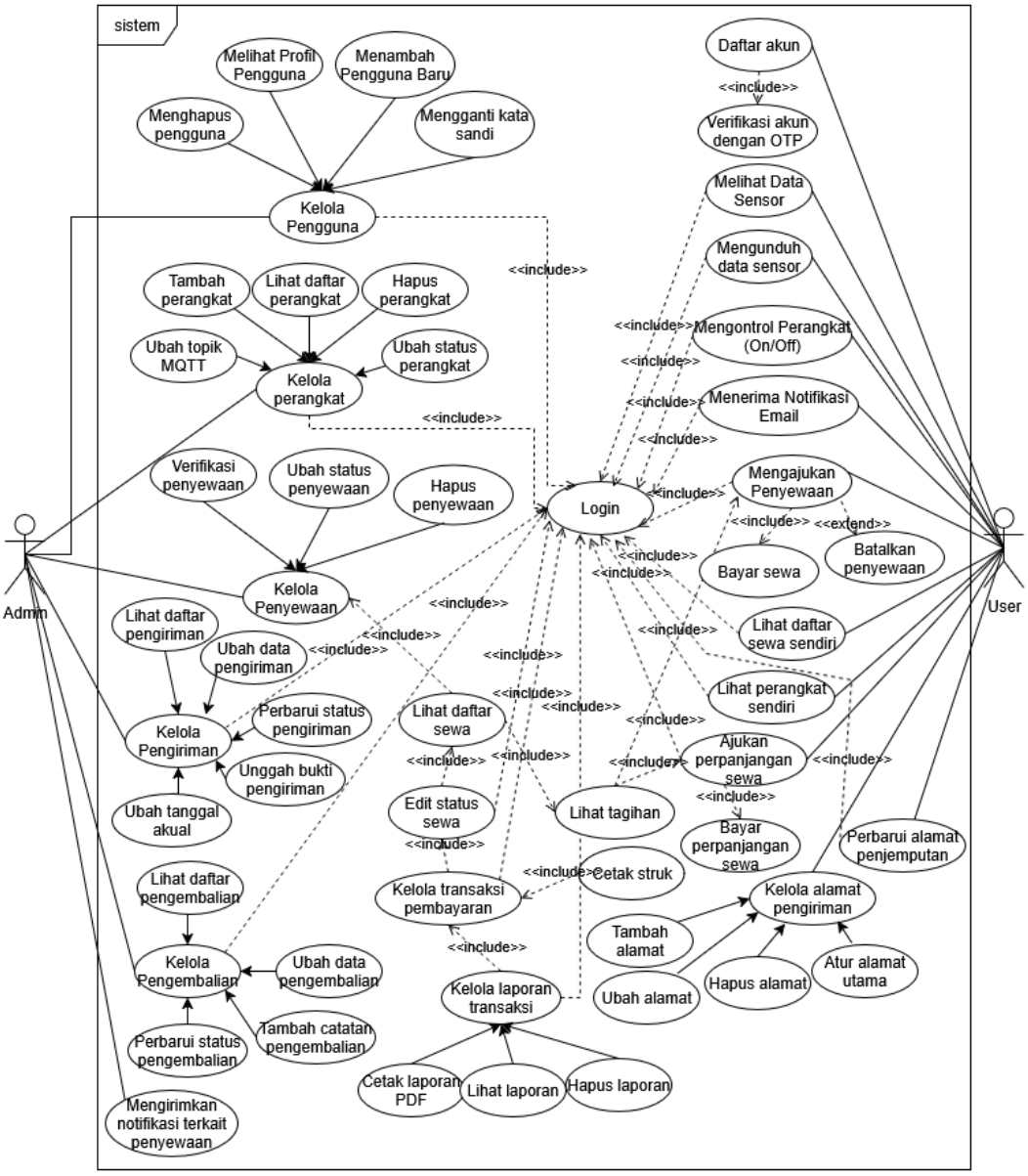
c. Keamanan Sistem

Keamanan sistem menjadi prioritas utama untuk menjaga integritas data dan mencegah akses tidak sah. Langkah-langkah yang diterapkan meliputi:

1) Semua komunikasi antara *frontend* dan *backend* dilindungi dengan HTTPS untuk mencegah intersepsi data.
2) Pengguna harus masuk terlebih dahulu untuk mendapatkan token autentikasi *JSON Web Token* (JWT), memastikan hanya pengguna yang terverifikasi dapat mengakses sistem.

2. Usecase

Setelah menjelaskan arsitektur sistem secara menyeluruh sebelumnya, langkah selanjutnya adalah menjabarkan fungsionalitas utama sistem dari perspektif pengguna. Implementasi sistem divisualisasikan menggunakan diagram *use case*, yang menunjukkan peran serta interaksi antara aktor dengan sistem. Diagram ini membantu dalam mengidentifikasi fitur-fitur yang harus didukung oleh sistem dan menjadi dasar dalam perancangan komponen *backend* serta alur komunikasi antar komponen.



Gambar 3.4. *Usecase Diagram*

Berdasarkan gambar 3.4, pengguna dalam sistem *backend* terdiri dari dua peran utama, yaitu *User* (pelanggan) dan *Admin*, yang masing-masing memiliki hak akses dan fungsi yang berbeda.

1) *User*/pelanggan

User adalah pihak yang menggunakan layanan penyewaan perangkat *rover* IoT. *User* dapat melakukan pendaftaran akun dan melakukan verifikasi menggunakan kode OTP. Setelah berhasil *login*, *user* dapat mengelola alamat pengiriman dengan menambah, mengubah, menghapus alamat, serta

menetapkan alamat utama. *User* juga dapat mengajukan penyewaan perangkat, membayar sewa, atau membatalkan pengajuan apabila statusnya masih dalam proses. Selain itu, user dapat melihat daftar penyewaan miliknya, termasuk status penyewaan, informasi tagihan, dan detail pengiriman. *User* juga memiliki akses untuk mengajukan perpanjangan sewa dan melakukan pembayarannya. Dalam hal pemantauan perangkat, user dapat mengontrol perangkat secara jarak jauh (menghidupkan atau mematikan), melihat data sensor yang dikirimkan perangkat, serta mengunduh dataset tersebut. *User* menerima notifikasi otomatis melalui email terkait penyewaan, pembayaran, pengiriman, maupun pengembalian perangkat.

2) Admin

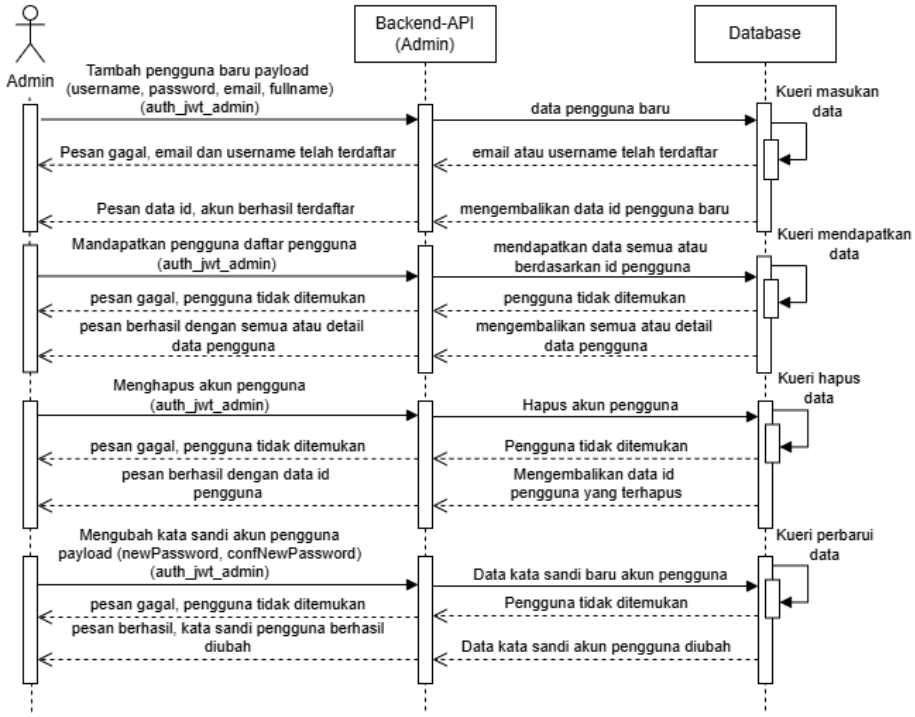
Admin adalah pengguna dengan hak akses penuh untuk mengelola seluruh bagian sistem. *Admin* dapat melakukan *login* dan mengakses fitur manajemen pengguna, seperti menambah akun baru, melihat profil pengguna, menghapus akun, dan mengganti kata sandi. *Admin* juga dapat mengelola perangkat IoT, mencakup penambahan perangkat, melihat daftar perangkat, menghapus perangkat, mengubah status perangkat, serta memperbarui topik MQTT untuk komunikasi sensor dan kontrol. Dalam pengelolaan penyewaan, admin bertugas memverifikasi pengajuan sewa, mengubah status penyewaan, dan menghapus data sewa jika diperlukan. *Admin* juga menangani manajemen pengiriman, mulai dari melihat daftar pengiriman, mengubah data pengiriman, memperbarui tanggal pengiriman, hingga mengunggah bukti pengiriman. Untuk proses pengembalian perangkat, *admin* dapat melihat daftar pengembalian, memperbarui status, mengubah data pengembalian, dan menambahkan catatan terkait. *Admin* juga bertanggung jawab untuk mengirimkan notifikasi terkait penyewaan kepada *user*. Selain itu, *admin* mengelola transaksi pembayaran, melihat dan memperbarui status sewa, mencetak struk, serta menyusun laporan transaksi. Laporan dapat dilihat, dicetak dalam bentuk PDF, maupun dihapus jika sudah tidak relevan.

3. Sequence Diagram

Setelah penyusunan *Use Case Diagram*, langkah selanjutnya yaitu menggambarkan alur interaksi aktor dengan sistem dalam *Sequence Diagram*

(diagram urutan interaksi). *Sequence Diagram* berfungsi untuk menggambarkan secara rinci alur komunikasi antar objek dalam sistem berdasarkan skenario tertentu.

a. Diagram *Sequence* Proses Manajemen Akun oleh Admin

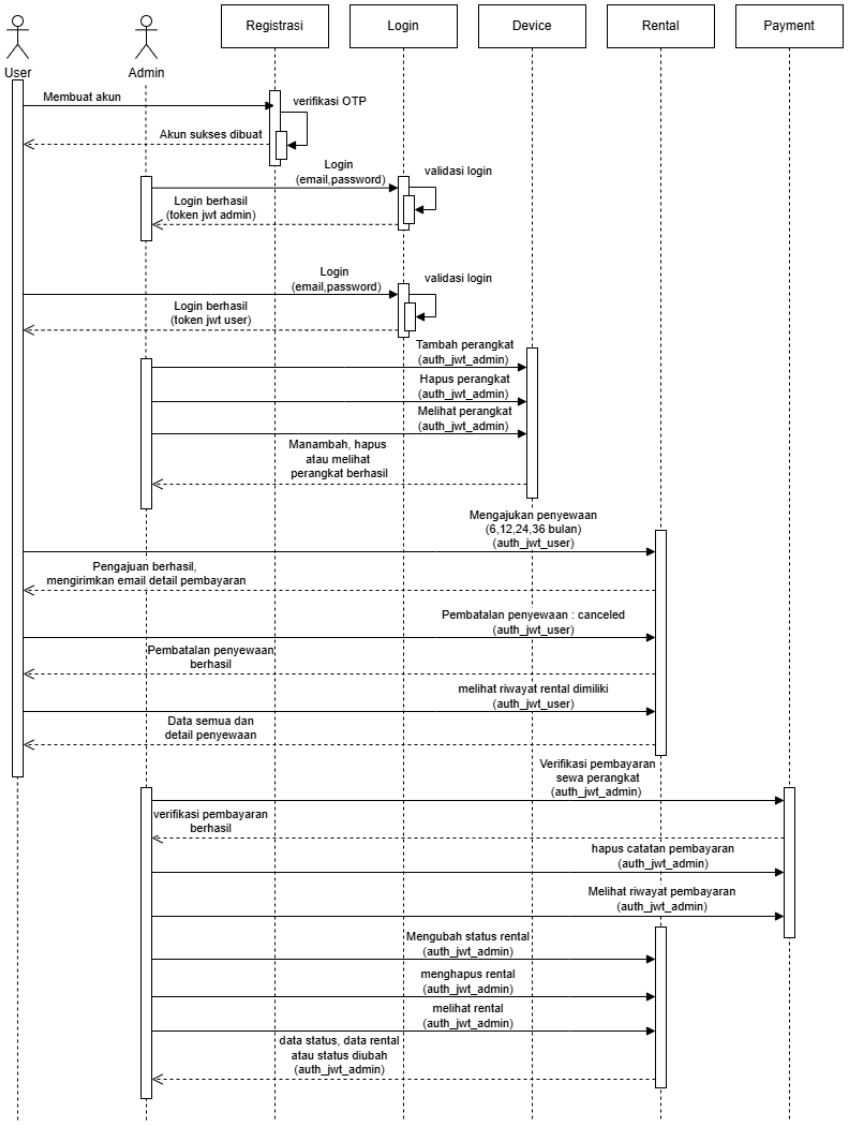


Gambar 3. 5. *Diagram Sequence* Proses Manajemen Akun

Gambar 3.5 menunjukkan proses manajemen akun oleh admin, yang mencakup penambahan pengguna, pengambilan daftar pengguna, penghapusan akun, dan perubahan kata sandi. *Admin* dapat menambahkan pengguna baru dengan mengirimkan data ke *Backend-API*, yang kemudian memverifikasi keberadaan email atau *username* di *database*. Jika sudah terdaftar, sistem menolak permintaan, sedangkan jika belum, data disimpan dan akun berhasil dibuat. *Admin* juga dapat mengambil daftar pengguna dari *database*. Jika data ditemukan, sistem mengembalikan informasi pengguna, jika tidak, sistem menampilkan pesan kesalahan. Untuk menghapus akun, *Backend-API* akan mengecek keberadaan pengguna di *database*, lalu menghapusnya jika ditemukan. Terakhir, admin bisa mengubah kata sandi pengguna. *Backend-API*

memverifikasi pengguna di *database* sebelum memperbarui kata sandi. Jika berhasil, sistem mengonfirmasi perubahan.

b. Diagram *Sequence* Proses Penyewaan Perangkat



Gambar 3. 6. *Diagram Sequence* Proses Penyewaan Perangkat

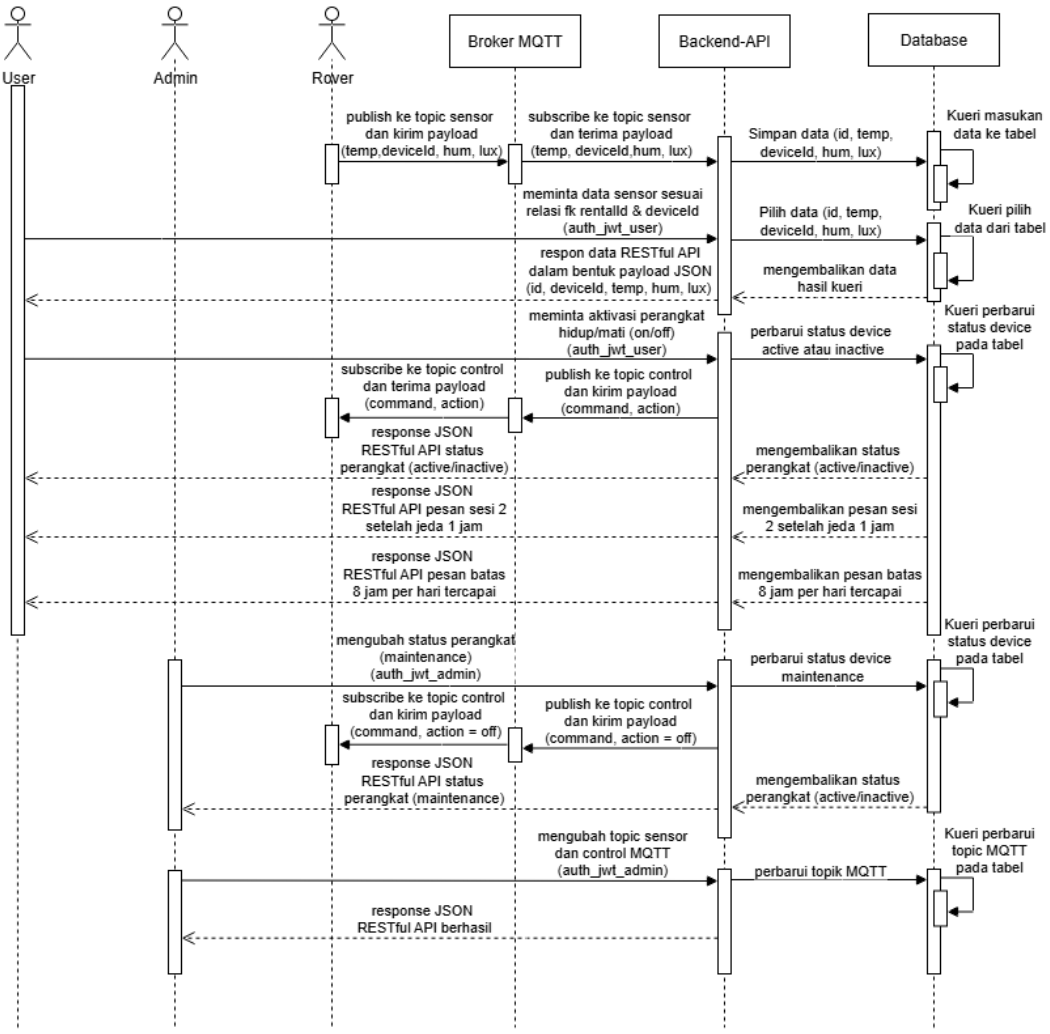
Gambar 3.6 menunjukkan alur proses penyewaan perangkat *rover* IoT oleh *user* serta peran *admin* dalam mengelola transaksi penyewaan. Proses dimulai dengan user melakukan registrasi akun, di mana sistem akan mengirimkan kode OTP untuk verifikasi sebelum akun berhasil dibuat. Setelah itu, baik *user* maupun *admin* dapat melakukan *login* dengan email dan *password* yang kemudian divalidasi oleh sistem. Jika *login* berhasil, sistem

akan memberikan token JWT yang digunakan untuk autentikasi dalam permintaan selanjutnya.

Setelah masuk, *admin* dapat menambah, menghapus, dan melihat daftar perangkat dalam sistem. *User* yang telah terautentikasi dapat mengajukan penyewaan perangkat dengan opsi durasi 6, 12, 24, atau 36 bulan. Jika pengajuan berhasil, sistem akan mengirimkan email berisi detail pembayaran kepada *user*. Selain itu, *user* juga memiliki opsi untuk membatalkan penyewaan sebelum pembayaran dikonfirmasi.

Pada tahap pembayaran, *admin* bertanggung jawab untuk memverifikasi transaksi penyewaan perangkat. Apabila pembayaran telah terkonfirmasi, sistem akan memperbarui status rental secara otomatis. *Admin* juga dapat melihat riwayat pembayaran *user* serta menghapus catatan pembayaran jika diperlukan. Selain itu, *admin* memiliki kewenangan untuk mengubah status penyewaan, menghapus data rental yang tidak valid, serta melihat daftar penyewaan yang sedang berlangsung.

c. *Diagram Sequence* Data Sensor dan Aktivasi Perangkat



Gambar 3. 7. *Diagram Sequence* Data Sensor dan Aktivasi Perang*kat*

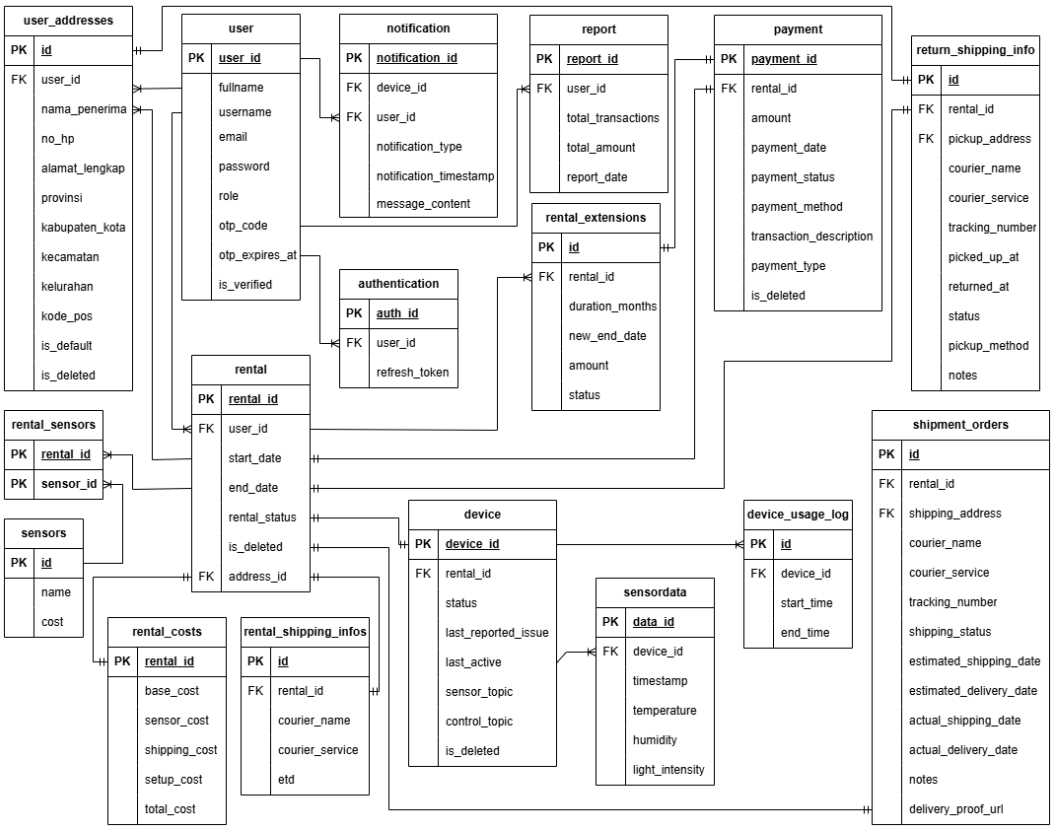
Gambar 3.7 menunjukkan diagram alur komunikasi data sensor dan kontrol perangkat dalam sistem penyewaan *rover* berbasis IoT. Proses diawali dengan perangkat rover yang mempublikasikan data sensor berupa suhu (*temperature*), kelembapan (*humidity*), dan intensitas cahaya (*lux*) ke topik tertentu di *Broker MQTT*. Broker kemudian meneruskan *payload* tersebut ke *Backend-API* yang telah berlangganan (*subscribe*) pada topik yang sama. Setelah menerima data, *Backend-API* menyimpan informasi tersebut ke dalam basis data, termasuk parameter seperti *id, deviceId, temperature, humidity*, dan *lux*. Data yang tersimpan ini selanjutnya dapat diminta oleh pengguna melalui antarmuka *RESTful API*, dan *Backend-API* akan merespons dalam format JSON berdasarkan relasi perangkat yang terdaftar pada penyewaan aktif.

Selain pemantauan sensor, diagram juga menggambarkan alur aktivasi dan kontrol perangkat oleh pengguna. Pengguna dengan peran user dapat mengirimkan permintaan untuk menghidupkan atau mematikan perangkat melalui *Backend-API*. Permintaan ini diteruskan ke *Broker MQTT* dalam bentuk *payload* yang berisi perintah (*command*) dan aksi (*action*). Perangkat akan merespons dan mengubah statusnya sesuai perintah yang diterima. Jika perangkat diaktifkan terlalu cepat setelah sesi sebelumnya, sistem akan menolak permintaan dan memberikan respons JSON bahwa sesi kedua hanya dapat dimulai setelah jeda minimal satu jam. Selain itu, sistem juga membatasi waktu operasional perangkat maksimal delapan jam per hari. Jika batas ini telah tercapai, permintaan akan ditolak dan dikembalikan dalam bentuk pesan JSON yang menyatakan bahwa batas penggunaan harian telah terpenuhi. Semua status perubahan yang valid disimpan kembali ke dalam basis data dan disampaikan ke pengguna sebagai respons.

Peran *admin* dalam sistem ini memiliki otoritas tambahan,yaitu mengubah status perangkat ke mode *maintenance* dan memperbarui konfigurasi topik MQTT. Akses tersebut dilakukan melalui permintaan yang telah diautentikasi oleh token JWT. Setelah *Backend-API* menerima permintaan dari admin, sistem akan meneruskannya ke Broker MQTT untuk dieksekusi oleh perangkat. Perubahan status atau konfigurasi yang berhasil dilakukan akan diperbarui ke dalam basis data, dan informasi keberhasilan dikembalikan kepada *admin* dalam bentuk respons JSON.

4. ERD *(Entity-Relationship Diagram)*

Sebelumnya telah dijelaskan bagaimana sistem beroperasi melalui *Sequence Diagram* yang menunjukkan interaksi komponen dalam merespons tindakan pengguna seperti penyewaan, pembayaran, dan pengiriman data sensor secara *real-time*, maka pada subbab ini akan dibahas struktur penyimpanan data melalui *Entity-Relationship Diagram* (ERD). Diagram ini memberikan gambaran mengenai entitas-entitas yang terlibat dalam sistem, atribut yang dimilikinya, serta relasi antar entitas tersebut. Penyusunan ERD sangat penting untuk merancang basis data yang konsisten, terstruktur, dan sesuai dengan kebutuhan sistem.



Gambar 3.8. *Entity-Relationship Diagram* (ERD)

ERD pada gambar 3.8 memuat delapan entitas utama: *user, rental, device, sensordata, notification, payment, authentication*, dan *report*. Dan juga terdapat beberapa entitas tambahan yang berfungsi mendukung proses penyewaan perangkat, pengiriman, hingga pengembalian perangkat. Entitas-entitas ini menunjukkan tingkat integrasi sistem yang tinggi dalam menangani alur penuh lifecycle perangkat. Berikut adalah penjelasan tiap entitas dan hubungan antar entitas:

1) *User* adalah entitas utama yang merepresentasikan pengguna sistem. Seorang pengguna dapat melakukan penyewaan perangkat (*rental*), menerima notifikasi (*notification*), melakukan pembayaran (*payment* melalui *rental*), dan memiliki sesi autentikasi (*authentication*).
2) *Rental* menyimpan informasi transaksi penyewaan perangkat yang dilakukan oleh pengguna. Hubungan ini bersifat *many-to-one*, yaitu banyak penyewaan dapat dilakukan oleh satu pengguna.

3) *Device* berisi informasi perangkat IoT yang disewakan. Setiap perangkat terkait dengan satu data penyewaan (*rental_id*) dan mengirimkan data sensor (*sensordata*).
4) *SensorData* merekam informasi lingkungan, meliputi suhu, kelembapan, serta intensitas cahaya yang diperoleh dari perangkat IoT. Setiap data sensor terhubung dengan satu perangkat (*device_id*).
5) *Notification* menyimpan pesan peringatan atau status sistem yang dikirim ke pengguna dan admin berdasarkan kondisi penyewaan dan perangkat. Hubungannya melibatkan *user_id* dan *device_id.*
6) *Payment* menyimpan catatan pembayaran berdasarkan data penyewaan. Satu penyewaan dapat memiliki satu atau lebih pembayaran.
7) *Authentication* menyimpan data autentikasi pengguna seperti *refresh_token* untuk keperluan manajemen sesi *login*.
8) *Report* digunakan oleh *admin* untuk merekap transaksi, jumlah penyewaan, dan total pembayaran selama periode tertentu. Entitas ini berelasi dengan entitas *user* dengan peran sebagai *admin*.
9) *Rental_Extensions* mencatat riwayat perpanjangan masa sewa perangkat. Setiap perpanjangan dikaitkan dengan satu *rental_id* dan menyimpan durasi tambahan, tanggal akhir baru, serta status verifikasi perpanjangan.
10) *Rental_Costs* menyimpan rincian biaya sewa yang dihitung per transaksi rental, meliputi *base_cost, sensor_cost, shipping_cost, setup_cost,* hingga *total_cost*. Ini memungkinkan sistem melakukan *breakdown* biaya secara transparan.
11) *Rental_Shipping_Infos* menyimpan informasi ekspedisi saat proses pengiriman perangkat ke penyewa, seperti nama kurir, layanan kurir, dan estimasi waktu pengiriman (ETD). Entitas ini berelasi langsung dengan *rental*.
12) *Return_Shipping_Info* menyimpan data pengembalian perangkat oleh pengguna, termasuk alamat pengambilan, jasa kurir, nomor resi, waktu penjemputan, status, dan catatan tambahan terkait pengembalian.

13) *Shipment_Orders* mencatat informasi logistik selama pengiriman awal perangkat ke pengguna, termasuk estimasi dan tanggal aktual pengiriman, status pengiriman, serta bukti pengiriman (dalam bentuk URL gambar).
14) *Device_Usage_Log* mencatat durasi penggunaan perangkat secara harian. Setiap entri merekam waktu mulai dan selesai penggunaan per hari berdasarkan *device_id.*
15) Rental_Sensors merupakan tabel perantara yang dirancang untuk menghubungkan entitas *rental* dan *sensor* dalam relasi *many-to-many*. Ini berarti satu penyewaan dapat menyertakan beberapa jenis sensor, dan satu jenis sensor bisa digunakan di banyak penyewaan. Tabel ini memiliki *primary key* gabungan dari *rental_id* dan *sensor_id.*
16) *Sensors* adalah entitas referensi yang mendefinisikan daftar jenis sensor yang tersedia, lengkap dengan nama dan biaya. Sensor ini dapat digunakan untuk menghitung *sensor_cost* pada *rental_costs*.
17) *User_Addresses* menyimpan daftar alamat milik pengguna yang digunakan untuk pengiriman perangkat sewa. Setiap entri memuat informasi alamat secara lengkap, mencakup nama penerima, nomor telepon, detail alamat, kelurahan, kecamatan, kota/kabupaten, provinsi, serta kode pos. Hubungan dengan entitas *user* bersifat *many-to-one*, yaitu satu pengguna dapat memiliki banyak alamat. Setiap penyewaan (*rental*) akan mengacu pada satu alamat melalui *address_id.*

Diagram ini menggambarkan bahwa sistem telah dirancang untuk mendukung integrasi menyeluruh mulai dari data pengguna, proses penyewaan dan perpanjangan, pengelolaan perangkat IoT, pencatatan data sensor secara berkala, hingga proses logistik pengiriman dan pengembalian perangkat serta pelaporan administratif. Selanjutnya, kamus data berfungsi untuk menjelaskan setiap atribut dalam entitas yang terdapat pada ERD. Penjelasan ini mencakup nama atribut, tipe data, dan deskripsi fungsionalnya. Berikut adalah kamus data untuk masing-masing entitas:

a) Kamus Data Entitas *User*

Kamus data untuk entitas *user* menjelaskan detail setiap atribut yang dimiliki oleh entitas tersebut. Informasi lengkap mengenai kamus data entitas *user* disajikan pada Tabel 3.3.

Tabel 3. 3. Kamus Data Entitas *User*

Field	**Tipe Data**	**Panjang Data**	**Keterangan**
id	*varchar*	22	ID unik menggunakan *nanoid* sepanjang 22 karakter.
username	*string*	50	Maksimal 50 karakter untuk efisiensi dan konsistensi.
fullname	*text*	-	Teks bebas untuk nama lengkap tanpa batasan khusus.
email	*text*	-	Email bervariasi panjangnya, disimpan sebagai teks.
password	*text*	-	*Hash password* disimpan dalam teks terenkripsi.
role	*enum*	-	Hanya menerima *admin* atau *user*.
otp_code	*varchar*	6	Kode OTP 6 digit sesuai standar umum.
otp_expires_at	*timestamp*	-	Menyimpan waktu kedaluwarsa OTP.
is_verified	*boolean*	-	Menunjukkan status verifikasi pengguna.

b) Kamus Data Entitas *Device*

Kamus data untuk entitas *device* menjelaskan detail setiap atribut yang dimiliki oleh entitas tersebut. Informasi lengkap mengenai kamus data entitas *device* disajikan pada Tabel 3.4.

Tabel 3.4. Kamus Data Entitas *Device*

Field	**Tipe Data**	**Panjang Data**	**Keterangan**
id	*varchar*	23	ID unik perangkat, panjang cukup untuk kombinasi *prefix* dan *nanoid*.
rental_id	*varchar*	23	Mengacu pada ID sewa, disamakan panjangnya agar konsisten.

Field	Tipe Data	Panjang Data	Keterangan
status	*enum*	-	status perangkat (*inactive, active*, *error*)
last_reported_issue	*timestamp*	-	Mencatat waktu terakhir masalah dilaporkan.
last_active	*integer*	-	Menyimpan durasi aktif terakhir dalam detik.
sensor_topic	*varchar*	255	Panjang 255 untuk fleksibilitas penamaan topik MQTT.
control_topic	*varchar*	255	Disamakan dengan *sensor_topic* demi konsistensi MQTT.
is_deleted	*boolean*	-	Menandai status *soft delete* tanpa menghapus data fisik.

c) Kamus Data Entitas *Rental*

Kamus data untuk entitas *rental* menjelaskan detail setiap atribut yang dimiliki oleh entitas tersebut. Informasi lengkap mengenai kamus data entitas *rental* disajikan pada Tabel 3.5.

Tabel 3.5. Kamus Data Entitas *Rental*

Field	Tipe Data	Panjang Data	Keterangan
rental_id	*varchar*	23	ID unik penyewaan, mengikuti format nanoid yang konsisten.
user_id	*varchar*	22	Mengacu pada ID pengguna, disesuaikan dengan sistem autentikasi.
address_id	*varchar*	15	ID alamat pengguna, cukup untuk ID pendek yang unik.
start_date	*timestamp*	-	Tanggal dan waktu mulai penyewaan.
end_date	*timestamp*	-	Tanggal dan waktu akhir penyewaan.
rental_status	*enum*	-	Status seperti p*ending, active, completed, awaiting-return* atau *canceled.*
is_deleted	*boolean*	-	Menandai apakah data telah dihapus secara logis (*soft delete*).

d) Kamus Data Entitas *Sensordata*

Kamus data untuk entitas *sensordata* menjelaskan detail setiap atribut yang dimiliki oleh entitas tersebut. Informasi lengkap mengenai kamus data entitas *sensordata* disajikan pada Tabel 3.6.

Tabel 3.6. Kamus Data Entitas *Sensordata*

Field	**Tipe Data**	**Panjang Data**	**Keterangan**
id	*varchar*	16	ID unik untuk setiap data sensor; 16 karakter cukup untuk menghindari duplikasi pada sistem dengan frekuensi tinggi.
device_id	*varchar*	23	Mengacu pada ID perangkat (*Device*), konsisten dengan panjang ID perangkat.
timestamp	*timestamp*	-	Menyimpan waktu lengkap pengambilan data untuk pelacakan historis.
temperature	*float*	-	Data suhu dari sensor (°C), disimpan dengan presisi desimal.
humidity	*float*	-	Data kelembaban (%), format float agar fleksibel menangani data sensor.
light_intensity	*float*	-	Data intensitas cahaya (lux), menggunakan float untuk fleksibilitas nilai tinggi.

e) Kamus Data Entitas *Report*

Kamus data untuk entitas *report* menjelaskan detail setiap atribut yang dimiliki oleh entitas tersebut. Informasi lengkap mengenai kamus data entitas *report* disajikan pada Tabel 3.7.

Tabel 3.7. Kamus Data Entitas *Report*

Field	**Tipe Data**	**Panjang Data**	**Keterangan**
id	*varchar*	23	*Primary key*, ID unik laporan; panjang 23 karakter digunakan agar konsisten dengan entitas lain dan cukup menampung kombinasi unik.

Field	Tipe Data	Panjang Data	Keterangan
user_id	*varchar*	22	*Foreign key* ke entitas *User*; mengikuti standar panjang ID pada tabel *users*.
total_transactions	*integer*	-	Total transaksi yang dirangkum dalam laporan; integer cukup karena hanya menghitung jumlah.
total_amount	*decimal(15, 2)*	-	Nilai total biaya dalam laporan; presisi 15 digit dengan 2 angka desimal agar akurat untuk kebutuhan keuangan.
report_date	*timestamp*	-	Tanggal dan waktu pembuatan laporan; memungkinkan pencatatan waktu presisi.

f) Kamus Data Entitas *Payment*

Kamus data untuk entitas *payment* menjelaskan detail setiap atribut yang dimiliki oleh entitas tersebut. Informasi lengkap mengenai kamus data entitas *payment* disajikan pada Tabel 3.8.

Tabel 3.8. Kamus Data Entitas Payment

Field	Tipe Data	Panjang Data	Keterangan
id	*varchar*	24	*Primary key*; ID unik pembayaran, panjang 24 karakter untuk menjamin keunikan global.
rental_id	*varchar*	23	*Foreign key* ke entitas Rental; konsisten dengan panjang ID pada tabel *rentals*.
amount	*decimal(15, 2)*	-	Jumlah pembayaran; presisi 15 digit dan 2 desimal sesuai kebutuhan transaksi finansial.
payment_date	*date*	-	Tanggal pembayaran dilakukan; cukup menggunakan tipe date karena hanya butuh hari, bukan jam.

Field	Tipe Data	Panjang Data	Keterangan
payment_status	*enum*	-	Status pembayaran: p*ending, success,* atau *failed;* memudahkan validasi status sistem.
payment_method	*varchar*	20	Nama metode pembayaran (misal: transfer, *e-wallet*); panjang 20 karakter cukup fleksibel.
transaction_ description	*text*	-	Deskripsi tambahan transaksi; text digunakan agar tidak terbatas panjang karakter.
payment_type	*varchar*	20	Jenis pembayaran, seperti inisial atau perpanjangan; cukup ditampung dalam 20 karakter.

g) Kamus Data Entitas *Notification*

Kamus data untuk entitas *notification* menjelaskan detail setiap atribut yang dimiliki oleh entitas tersebut. Informasi lengkap mengenai kamus data entitas *notification* disajikan pada Tabel 3.9.

Tabel 3. 9. Kamus Data Entitas *Notification*

Field	Tipe Data	Panjang Data	Keterangan
id	*varchar*	23	*Primary key*; ID unik untuk setiap notifikasi.
user_id	*varchar*	22	*Foreign key* ke entitas *users*; mengaitkan notifikasi dengan pengguna terkait.
notification_type	*enum*	-	Tipe notifikasi, seperti *alert, warning*, atau *info*; memudahkan kategorisasi.
notification_timestamp	*timestamp*	-	Waktu pengiriman notifikasi secara lengkap (tanggal dan jam).
message_content	*text*	-	Isi pesan; menggunakan text agar fleksibel menampung berbagai panjang pesan.

h) Kamus Data Entitas *Authentication*

Kamus data untuk entitas *authentication* menjelaskan detail setiap atribut yang dimiliki oleh entitas tersebut. Informasi lengkap mengenai kamus data entitas authentication disajikan pada Tabel 3.10.

Tabel 3.10. Kamus Data Entitas *Authentication*

Field	**Tipe Data**	**Panjang Data**	**Keterangan**
id	*varchar*	16	*Primary key*, identifikasi unik untuk setiap token autentikasi.
user_id	*varchar*	22	*Foreign key* yang menghubungkan token dengan entitas users.
token	*text*	-	Refresh token JWT untuk memperpanjang sesi autentikasi.
created_at	*timestamp*	-	Waktu saat token dibuat.

i) Kamus Data Entitas User_addresses

Kamus data untuk entitas *user_addresses* menjelaskan detail setiap atribut yang dimiliki oleh entitas tersebut. Informasi lengkap mengenai kamus data entitas user_addresses disajikan pada Tabel 3.11.

Tabel 3. 11. Kamus Data Entitas *User_addresses*

Field	**Tipe Data**	**Panjang Data**	**Keterangan**
id	*varchar*	15	*Primary key*, identifikasi unik untuk setiap alamat pengiriman.
user_id	*varchar*	22	*Foreign key* yang menghubungkan alamat dengan pengguna terkait.
nama_penerima	*varchar*	100	Nama lengkap orang yang akan menerima pengiriman.
no_hp	*varchar*	20	Nomor *handphone* penerima untuk keperluan konfirmasi atau kurir.
alamat_lengkap	*text*	-	Detail lengkap alamat tujuan; fleksibel untuk panjang variatif.
provinsi	*varchar*	100	Nama provinsi tempat tujuan pengiriman.

Field	Tipe Data	Panjang Data	Keterangan
kabupaten_kota	*varchar*	100	Nama kabupaten atau kota tujuan pengiriman.
kecamatan	*varchar*	100	Nama kecamatan tujuan pengiriman.
kelurahan	*varchar*	100	Nama kelurahan atau desa tujuan pengiriman.
kode_pos	*varchar*	5	Kode pos sesuai wilayah alamat pengiriman.
is_default	*boolean*	-	Penanda apakah alamat ini merupakan alamat utama pengguna.
is_deleted	*boolean*	-	Penanda apakah alamat ini sudah dihapus secara *soft delete* dari sistem.

j) Kamus Data Entitas Rental_sensors

Kamus data untuk entitas *rental_sensors* menjelaskan detail setiap atribut yang dimiliki oleh entitas tersebut. Informasi lengkap mengenai kamus data entitas rental_sensors disajikan pada Tabel 3.12.

Tabel 3. 12. Kamus Data Entitas *Rentals_sensors*

Field	Tipe Data	Panjang Data	Keterangan
rental_id	varchar	23	*Foreign key* ke *rentals*; panjang disesuaikan dengan format ID sewa.
sensor_id	varchar	32	*Foreign key* ke *sensors*; panjang dipilih agar kompatibel kode sensor unik.

k) Kamus Data Entitas *Sensors*

Kamus data untuk entitas *sensors* menjelaskan detail setiap atribut yang dimiliki oleh entitas tersebut. Informasi lengkap mengenai kamus data entitas *sensors* disajikan pada Tabel 3.13.

Tabel 3. 13. Kamus Data Entitas *Sensors*

Field	Tipe Data	Panjang Data	Keterangan
id	*varchar*	32	*Primary key*; panjang disesuaikan agar kompatibel dengan format ID unik.

Field	Tipe Data	Panjang Data	Keterangan
name	*varchar*	64	Nama sensor; panjang cukup untuk menampung nama deskriptif.
cost	*integer*	-	Biaya sewa tambahan per sensor; minimal bernilai 0.

l) Kamus Data Entitas *Rental_costs*

Kamus data untuk entitas *rental_costs* menjelaskan detail setiap atribut yang dimiliki oleh entitas tersebut. Informasi lengkap mengenai kamus data entitas *rental_costs* disajikan pada Tabel 3.14.

Tabel 3. 14. Kamus Data Entitas *Rental_costs*

Field	Tipe Data	Panjang Data	Keterangan
rental_id	*varchar*	23	*Primary key*; ID unik penyewaan. Panjang 23 dipilih untuk mengakomodasi format prefiks dan ID dari tabel rentals.
base_cost	*integer*	-	Biaya dasar penyewaan perangkat tanpa tambahan fitur.
sensor_cost	*integer*	-	Biaya tambahan dari sensor yang disewa, nilai minimal 0.
shipping_cost	*integer*	-	Biaya pengiriman perangkat ke alamat penyewa, nilai minimal 0.
setup_cost	*integer*	-	Biaya pemasangan atau instalasi perangkat di lokasi penyewa, nilai minimal 0.
total_cost	*integer*	-	Total keseluruhan biaya sewa, merupakan hasil penjumlahan seluruh komponen biaya.

m) Kamus Data Entitas *Rental_shipping_infos*

Kamus data untuk entitas *rental_shipping_infos* menjelaskan detail setiap atribut yang dimiliki oleh entitas tersebut. Informasi lengkap mengenai kamus data entitas *rental_shipping_infos* disajikan pada Tabel 3.15.

Tabel 3. 15. Kamus Data Entitas *Rental_shipping_infos*

Field	Tipe Data	Panjang Data	Keterangan
id	*varchar*	25	*Primary key*; ID unik informasi pengiriman. Panjang 25 disesuaikan dengan format ID.
rental_id	*varchar*	23	*Foreign key* ke tabel *rentals*. Panjang konsisten dengan *rental_id* pada entitas terkait.
courier_name	*text*	-	Nama ekspedisi yang digunakan, fleksibel untuk berbagai nama penyedia layanan.
courier_service	*text*	-	Jenis layanan pengiriman yang digunakan.
etd	*integer*	-	Estimasi lama pengiriman (dalam satuan hari).

n) Kamus Data Entitas *Shipment_orders*

Kamus data untuk entitas *shipment_orders* menjelaskan detail setiap atribut yang dimiliki oleh entitas tersebut. Informasi lengkap mengenai kamus data entitas *shipment_orders* disajikan pada Tabel 3.16.

Tabel 3. 16. Kamus Data Entitas *Shipment_orders*

Field	Tipe Data	Panjang Data	Keterangan
id	*varchar*	25	*Primary key*; ID unik untuk setiap pesanan pengiriman. Panjang 25 cukup untuk format ID khusus.
rental_id	*varchar*	23	*Foreign key* ke *rentals*. Panjang disesuaikan dengan ID penyewaan.
shipping_address_id	*varchar*	15	*Foreign key* ke *user_addresses*. Panjang mencukupi untuk format ID pendek.
courier_name	*text*	-	Nama ekspedisi, fleksibel agar mendukung berbagai nama jasa kirim.
courier_service	*text*	-	Jenis layanan pengiriman seperti REG, YES dan *Sameday*.

Field	Tipe Data	Panjang Data	Keterangan
tracking_number	*text*	-	Nomor resi pelacakan, bisa kosong awalnya.
shipping_status	*enum*	-	Status pengiriman: *waiting, shipped*, atau *delivered*, *default waiting*.
estimated_shipping_date	*timestamp*	-	Perkiraan tanggal barang dikirim.
estimated_delivery_date	*timestamp*	-	Perkiraan tanggal diterima.
actual_shipping_date	*timestamp*	-	Tanggal sebenarnya barang dikirimkan.
actual_delivery_date	*timestamp*	-	Tanggal sebenarnya barang diterima.
notes	*text*	-	Catatan tambahan pengiriman.
delivery_proof_url	*text*	-	URL bukti pengiriman (foto/dokumen digital).

o) Kamus Data Entitas *Device_usage_log*

Kamus data untuk entitas *device_usage_log* menjelaskan detail setiap atribut yang dimiliki oleh entitas tersebut. Informasi lengkap mengenai kamus data entitas d*evice_usage_log* disajikan pada Tabel 3.17.

Tabel 3. 17. Kamus Data Entitas *Device_usage_log*

Field	Tipe Data	Panjang Data	Keterangan
id	*varchar*	11	*Primary key*; ID unik untuk setiap catatan penggunaan perangkat. Panjang 11 cukup untuk format ID pendek.
device_id	*varchar*	23	*Foreign key* ke entitas *devices*; menghubungkan log dengan perangkat terkait.
start_time	*timestamp*	-	Waktu mulai penggunaan perangkat.
end_time	*timestamp*	-	Waktu akhir penggunaan. Bisa *null* jika masih aktif.

p) Kamus Data Entitas *Rental_extensions*

Kamus data untuk entitas *rental_extensions* menjelaskan detail setiap atribut yang dimiliki oleh entitas tersebut. Informasi lengkap mengenai kamus data entitas *rental_extensions* disajikan pada Tabel 3.18.

Tabel 3. 18. Kamus Data Entitas *Rental_extensions*

Field	**Tipe Data**	**Panjang Data**	**Keterangan**
id	*varchar*	15	*Primary key*; ID unik setiap perpanjangan. Panjang 15 cukup untuk format ID ringkas *nanoid*.
rental_id	*varchar*	23	*Foreign key* ke entitas *rentals*; mengaitkan dengan penyewaan utama.
duration_months	*integer*	-	Lama perpanjangan dalam bulan (6, 12, 24, atau 36).
new_end_date	*date*	-	Tanggal berakhir baru setelah perpanjangan.
amount	*integer*	-	Biaya perpanjangan (harus positif).
status	*varchar*	50	Status perpanjangan: *pending_payment, completed,* atau *failed*. Panjang 50 mengakomodasi *enum* yang eksplisit.

q) Kamus Data Entitas *Return_shipping_info*

Kamus data untuk entitas *return_shipping_info* menjelaskan detail setiap atribut yang dimiliki oleh entitas tersebut. Informasi lengkap mengenai kamus data entitas *return_shipping_info* disajikan pada Tabel 3.19.

Tabel 3. 19. Kamus Data Entitas *Return_shipping_info*

Field	**Tipe Data**	**Panjang Data**	**Keterangan**
id	*varchar*	25	*Primary key*; ID unik pengembalian. Panjang 25 mencukupi format ID standar *nanoid*.
rental_id	*varchar*	23	*Foreign key* ke *rentals*; menghubungkan dengan data penyewaan.

Field	Tipe Data	Panjang Data	Keterangan
pickup_address_id	*varchar*	15	*Foreign key* ke *user_addresses*; *nullable* jika alamat dihapus.
courier_name	*text*	-	Nama ekspedisi (fleksibel karena bisa bervariasi panjangnya).
courier_service	*text*	-	Layanan pengiriman, misal *JNE REG, SICEPAT BEST,* dan lain-lain.
tracking_number	*text*	-	Nomor resi pengembalian; menggunakan text karena bisa alfanumerik variatif.
picked_up_at	*timestamp*	-	Waktu perangkat dijemput.
returned_at	*timestamp*	-	Waktu perangkat diterima kembali.
status	*enum*	-	Status pengembalian: *requested, returning, returned* dan *failed.*
pickup_method	*varchar*	20	Metode pengembalian: *pickup* atau *dropoff*. Panjang 20 cukup untuk *enum* eksplisit.
notes	*text*	-	Catatan admin jika ada.

Tanda "-" berarti panjang tidak ditentukan karena tipe datanya tidak memerlukannya. Misalnya *text, timestamp, boolean*, dan *enum* otomatis ditangani oleh sistem basis data tanpa batasan panjang manual. Panjang karakter ditentukan berdasarkan kebutuhan data yang wajar, efisiensi penyimpanan, serta standar umum seperti panjang nanoid (22–23), batas input pengguna (50), dan fleksibilitas komunikasi MQTT (255).

5. Perancangan Mekanisme *Subscription-Based Rental*

Sistem ini dirancang menggunakan pendekatan *subscription-based rental*, di mana pengguna dapat memilih durasi penyewaan perangkat dalam kelipatan bulanan, yaitu 6, 12, 24, atau 36 bulan. Setiap pilihan durasi memiliki skema diskon tersendiri untuk mendorong penyewaan jangka panjang. Perhitungan biaya sewa dilakukan berdasarkan rumus berikut:

$$Total\ Hari = Durasi\ Bulan \times 30$$

$$Biaya\ Dasar = Total\ Hari \times R$$

$$Diskon = Biaya\ Dasar \times D$$

$$Biaya\ Akhir = Biaya\ Dasar - Diskon$$

Dengan:

R = tarif harian (Rp100.000)

D = persentase diskon (6 bulan = 0,05; 12 bulan = 0,10; 24 bulan = 0,15; 36 bulan = 0,20)

Biaya akhir dicatat sebagai *baseCost*, dan total biaya penyewaan dihitung berdasarkan beberapa komponen yang ditampilkan pada tabel 3.20.

Tabel 3. 20. Komponen Biaya Penyewaan

Komponen	**Keterangan**
baseCost	Biaya pokok sewa berdasarkan durasi langganan dan diskon otomatis.
sensorCost	Biaya tambahan berdasarkan akumulasi harga dari sensor-sensor yang dipilih.
shippingCost	Biaya pengiriman dan pengembalian perangkat ke lokasi tujuan, diperoleh dari API eksternal Raja Ongkir.
setupCost	Biaya pemasangan awal perangkat, bersifat tetap sebesar Rp1.000.000.

Formula total biaya sewa yang digunakan:

$$totalCost = baseCost + sensorCost + shippingCost + setupCost$$

Untuk perpanjangan masa sewa, sistem hanya menghitung baseCost saja tanpa memasukkan ulang biaya sensor, pengiriman, atau pemasangan. Hal ini dilakukan karena perangkat dan sensor sudah berada di lokasi pengguna, sehingga tidak ada biaya tambahan selain pokok langganan.

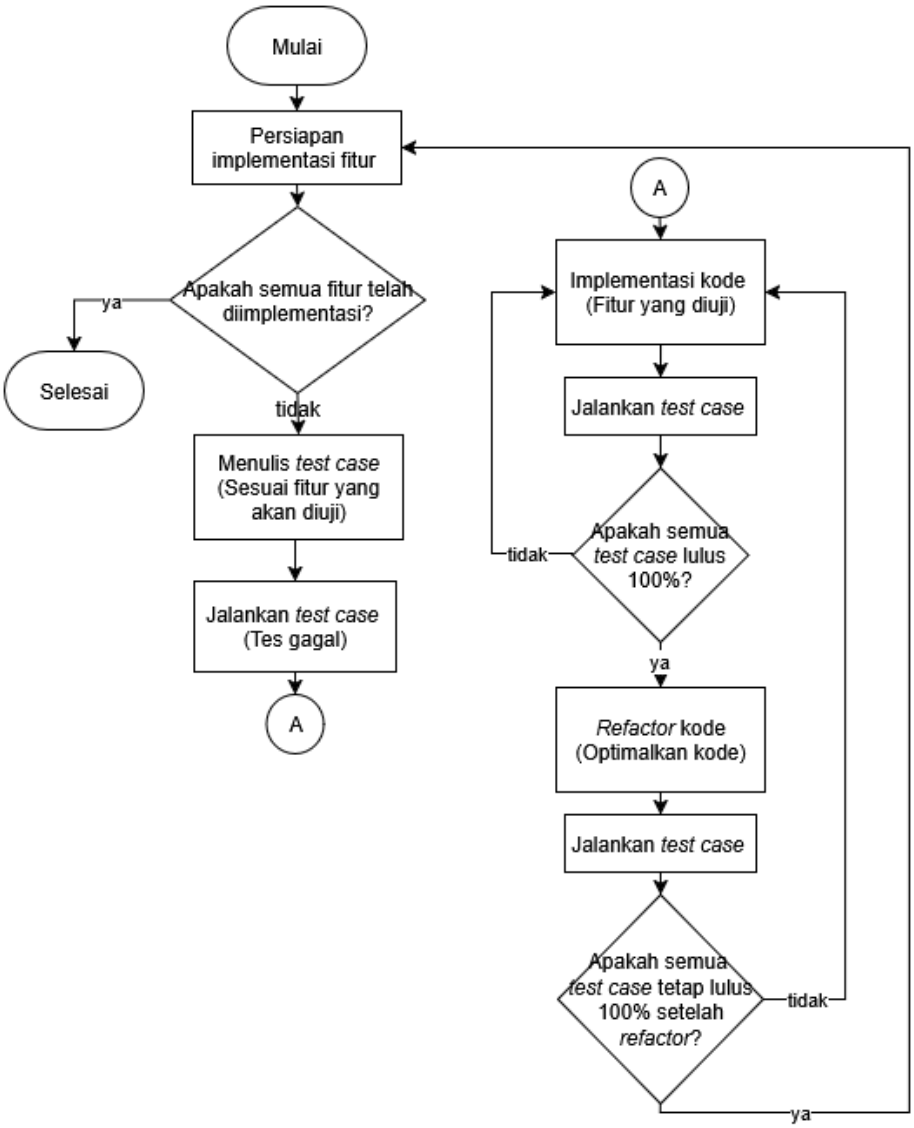
3.3.2.3 Penulisan Kode

Pembuatan sistem backend menggunakan *framework Express.js* dengan menggunakan bahasa pemrograman *Node.js.* Pengkodean *backend* digunakan untuk memfungsikan tampilan yang dibuat *frontend*.

3.3.2.4 Pengujian Sistem

a. *Test-Driven Development*

Tahap pengujian sistem dilakukan dengan pendekatan *Test-Driven Development* (TDD) guna memastikan setiap komponen *backend* berfungsi sesuai spesifikasi yang telah ditentukan. Metode ini memiliki siklus yang jelas yang melibatkan tiga langkah utama: *Red*, *Green*, dan *Refactor*.



Gambar 3.9. Diagram Alir Implementasi TDD Pada Pengembangan Aplikasi *Backend*

Berdasarkan gambar 3.9, pada metode *Test-Driven Development* (TDD), proses dimulai dengan mengevaluasi apakah seluruh fitur telah diimplementasikan sesuai spesifikasi. Jika semua fitur telah selesai, maka proses pengujian dianggap selesai. Namun, jika masih terdapat fitur yang belum diimplementasikan, langkah pertama yang dilakukan adalah menuliskan *test case* untuk fitur tersebut. *Test case* ini mencakup skenario pengujian berupa *input, output,* dan perilaku sistem yang diharapkan.

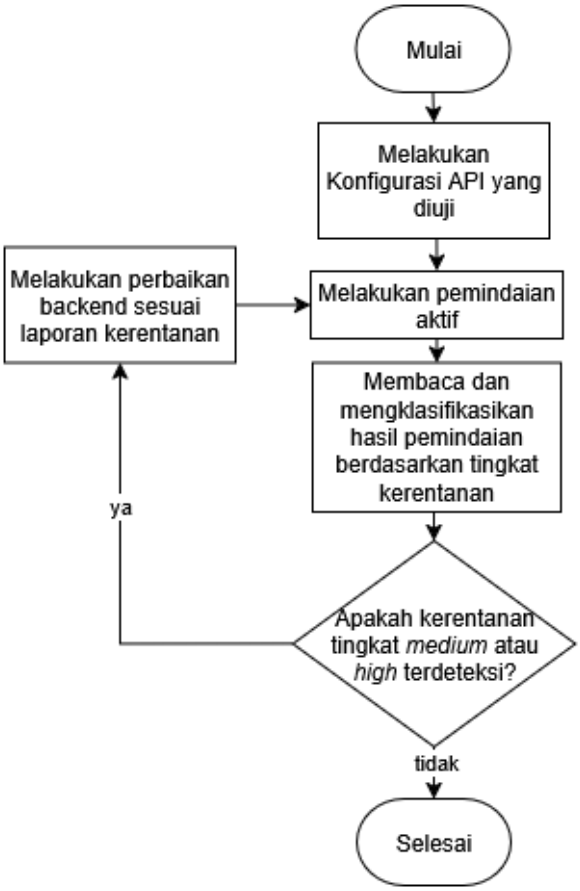
Setelah *test case* ditulis, pengujian dijalankan. Pada tahap awal ini, pengujian akan gagal (ditandai warna merah) karena kode fungsionalitas belum tersedia. Langkah selanjutnya mencakup implementasi kode perangkat lunak agar sesuai dengan skenario yang didefinisikan dalam *test case*. Setelah implementasi selesai, pengujian dijalankan kembali. Jika seluruh *test case* berhasil dijalankan tanpa *error* (ditandai warna hijau), maka proses dilanjutkan ke tahap *refactoring*.

Proses *refactoring* bertujuan meningkatkan kualitas struktur kode tanpa memengaruhi fungsi atau perilakunya. Sesudah perubahan, semua *test case* yang ada dieksekusi ulang untuk memverifikasi bahwa tidak muncul regresi. Jika hasil pengujian masih menunjukkan keberhasilan (warna hijau), maka iterasi dianggap selesai dan proses dapat berlanjut ke fitur berikutnya.

Proses ini diulang secara berkelanjutan untuk setiap fitur sistem. Indikator keberhasilan dalam pendekatan TDD adalah ketika seluruh *test case* lulus tanpa *error*, dengan tingkat keberhasilan 100%, menandakan bahwa implementasi telah sesuai dengan spesifikasi dan bebas dari kesalahan fungsional.

b. Pengujian Kerentanan

Untuk pengujian kerentanan aplikasi, penelitian ini memanfaatkan OWASP ZAP (*Zed Attack Proxy*) sebagai alat utama dalam melakukan pemindaian dan analisis potensi kerentanan pada sistem backend.



Gambar 3. 10. Pengujian Kerentanan menggunakan OWASP ZAP

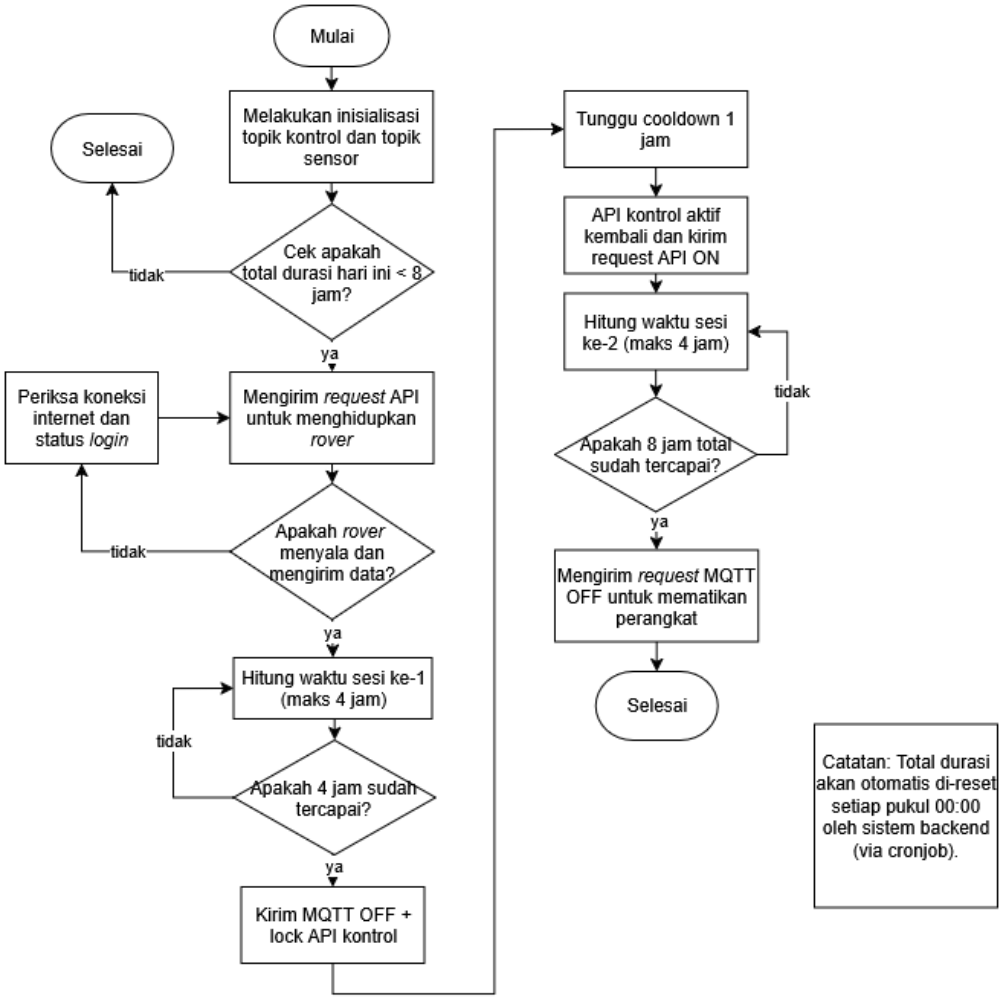
Berdasarkan gambar 3.10, proses pengujian dimulai dengan mengonfigurasi ZAP agar dapat mengakses API yang akan diuji, kemudian dilanjutkan dengan menjalankan pemindaian menggunakan metode pemindaian aktif (*active scan*) terhadap seluruh *endpoint* publik. ZAP secara otomatis mencoba berbagai skenario serangan berdasarkan daftar OWASP Top 10. Setelah proses pemindaian selesai, ZAP menghasilkan laporan yang memuat daftar kerentanan yang ditemukan, lengkap dengan klasifikasi tingkat risiko*: Informational, Low, Medium*, dan *High*, berdasarkan dampaknya terhadap sistem [33].

Langkah selanjutnya adalah membaca dan mengklasifikasikan hasil pemindaian berdasarkan tingkat kerentanan. Jika ditemukan kerentanan dengan tingkat risiko *Medium* atau *High*, maka sistem *backend* diperbaiki sesuai dengan rekomendasi mitigasi yang disertakan dalam laporan. Setelah dilakukan perbaikan, proses pemindaian diulang untuk memastikan bahwa kerentanan tersebut telah diatasi.

Indikator keberhasilan pengujian kerentanan adalah tidak ditemukannya kerentanan dengan tingkat risiko *Medium* atau *High* pada hasil *active scan*. Dengan demikian, jika seluruh temuan berada pada level risiko *Low* atau *Informational*, maka sistem dinyatakan telah memenuhi standar keamanan minimum.

c. Pengujian Komunikasi MQTT

Pengujian komunikasi MQTT dilakukan untuk memastikan perangkat dapat mengirim dan menerima data secara fungsional. Postman digunakan untuk mengirim request API yang mengontrol aktivasi dan deaktivasi perangkat, lalu diverifikasi melalui broker MQTT. Sistem *rover* dibatasi maksimal 8 jam operasional per hari, sesuai jam kerja normal di lapangan, guna mencegah penggunaan berlebihan (*overuse*).



Gambar 3. 11. Diagram Alur Pengujian Komunikasi MQTT

Berdasarkan gambar 3.11, pengujian komunikasi MQTT diawali dengan menyiapkan sistem dan menginisialisasi topik kontrol serta topik sensor pada *backend* dan perangkat *rover*. Inisialisasi ini bertujuan untuk memastikan komunikasi dua arah antara perangkat dan server dapat berjalan melalui topik MQTT yang telah ditentukan.

Langkah berikutnya adalah melakukan pengecekan terhadap total durasi penggunaan perangkat pada hari tersebut. Jika durasi penggunaan masih kurang dari 8 jam, maka permintaan API dikirim menggunakan *Postman* untuk mengaktifkan *rover*. Sebelum perangkat dihidupkan, sistem juga memverifikasi koneksi internet dan status *login* pengguna. Jika koneksi atau status *login* gagal, pengujian dihentikan dan pengguna diminta untuk memeriksa ulang jaringan atau kredensial.

Jika koneksi dan login berhasil, sistem kemudian memverifikasi apakah *rover* menyala dan mulai mengirimkan data sensor melalui topik MQTT. Jika perangkat menyala dan data berhasil dikirim, maka sistem akan menghitung waktu sesi ke-1 dengan durasi maksimum 4 jam. Setelah waktu 4 jam sesi pertama tercapai, sistem secara otomatis mengirim perintah MQTT untuk mematikan perangkat dan mengunci API kontrol agar tidak dapat digunakan selama masa *cooldown*. Sistem kemudian menunggu masa cooldown selama 1 jam.

Setelah masa cooldown berakhir, API kontrol diaktifkan kembali secara otomatis, dan sistem kembali mengirim *request* API untuk menghidupkan *rover*. Sesi ke-2 dimulai dengan durasi maksimum 4 jam. Sistem akan terus menghitung waktu sesi ke-2 hingga total penggunaan perangkat mencapai 8 jam dalam satu hari. Jika total durasi 8 jam tercapai, sistem akan secara otomatis mengirim perintah MQTT OFF untuk mematikan perangkat, menandakan bahwa batas maksimum penggunaan harian telah tercapai.

Jika pada proses pengiriman atau penerimaan data terjadi kegagalan, sistem akan kembali melakukan pemeriksaan koneksi internet dan status *login*. Pengujian dinyatakan selesai apabila perangkat telah digunakan secara penuh selama 8 jam (terbagi dalam dua sesi dengan jeda 1 jam) dan berhasil dimatikan secara otomatis oleh sistem.

3.3.2.5 *Deployment* Aplikasi *Backend*

Proses *deployment* aplikasi *backend* dilakukan secara otomatis menggunakan *GitHub Actions* sebagai layanan CI/CD (*Continuous Integration/Continuous Deployment*). Proses ini dirancang untuk memastikan bahwa aplikasi backend yang dideploy di server production selalu dalam kondisi yang siap digunakan dan stabil.

Gambar 3.12. Alur *Deployment* Aplikasi Backend di *Production*

Berdasarkan gambar 3.8, proses *deployment* aplikasi *backend* menggunakan *GitHub Actions* dimulai dengan pembuatan *file* konfigurasi CI/CD dalam format YAML yang disimpan di direktori *.github/workflows*. File ini mencakup tahapan otomatisasi, seperti menjalankan pengujian, membangun aplikasi, dan mengirimkan kode ke server *production*. Selanjutnya, konfigurasi *environment* server *Virtual Private Server* (VPS) dilakukan dengan menetapkan detail akses SSH, seperti *hostname, port, username*, dan *password*. Langkah ini memastikan *GitHub Actions* dapat terhubung ke server secara aman untuk menjalankan proses *deployment*.

Fitur baru dikembangkan di branch terpisah, lalu diunggah ke GitHub untuk diuji dan ditinjau sebelum digabung ke branch utama. Proses merge ini memicu *workflow* CI untuk memverifikasi kompatibilitas kode baru dan melakukan pengujian otomatis. Jika semua tahapan berhasil, *GitHub Actions* melanjutkan proses *Continuous Deployment* (CD), termasuk pengiriman *file* aplikasi terbaru ke *server* VPS dan *restart* layanan di *production*. Pendekatan ini meningkatkan efisiensi dan keandalan proses *deployment*, serta meminimalkan potensi kesalahan manual.

3.3.3 Penulisan Laporan

Tujuan dari penulisan laporan ini adalah untuk menyajikan hasil penelitian serta mendokumentasikan seluruh rangkaian kegiatan secara sistematis. Laporan ini bertujuan agar pembaca dapat memahami proses penelitian yang telah dilakukan dan dapat digunakan sebagai referensi untuk penelitian di masa mendatang.

V SIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil dan pembahasan yang telah dilakukan dalam penelitian dengan judul "Pengembangan *Backend* Sistem IoT Berbasis *Express.js* untuk Monitoring Lingkungan Dan Penyewaan *Rover* di Perkebunan Kelapa Sawit", maka dapat disimpulkan:

1. Sistem *backend* berhasil dikembangkan menggunakan pendekatan *Test-Driven Development* (TDD) dengan *Node.js* dan *Express.js*, menghasilkan 63 *endpoint* RESTful API yang telah teruji dan terdokumentasi lengkap.
2. Komunikasi *real-time* dua arah antara perangkat *rover* dan server berhasil diimplementasikan menggunakan protokol MQTT dengan fitur kontrol penggunaan harian (8 jam) dan sistem *cooldown* (1 jam) untuk mencegah *overuse* perangkat.
3. Kualitas sistem terbukti sangat baik dengan tingkat keberhasilan pengujian 100%:
    a. *Unit test*: 122 *test case* (100% *pass*)
    b. *Integration test*: 171 *test case* (100% *pass*)
    c. API *endpoint test*: 702 skenario (100% *pass*)
    d. *Security test*: Tidak ditemukan kerentanan kritis dari 47 plugin ZAP
4. Sistem *subscription-based rental* berhasil diimplementasikan dengan skema diskon progresif (5%-20%) berdasarkan durasi sewa (6-36 bulan), serta perhitungan biaya otomatis yang mencakup *base cost, sensor cost, shipping cost,* dan *setup cost.*
5. *Deployment* otomatis berhasil dilakukan menggunakan *CI/CD pipeline GitHub Actions* ke VPS dengan stabilitas tinggi, mendukung *Node.js* versi 20 dan 22 LTS.

5.2 Saran

Setelah melakukan penelitian tentang "Pengembangan *Backend* Sistem IoT Berbasis *Express.js* untuk Monitoring Lingkungan Dan Penyewaan *Rover* di Perkebunan Kelapa Sawit", beberapa saran diberikan sebagai berikut:

1. Perluasan fitur aplikasi seperti integrasi *payment gateway* akan mempermudah transaksi pembayaran sewa *rover*.
2. Mengoptimalkan performa sistem *backend*, misalnya dengan menerapkan teknik pembagian beban kerja (*load balancing*), pemanfaatan penyimpanan sementara (*caching*), serta arsitektur berbasis layanan kecil (*microservices*) untuk meningkatkan skalabilitas.
3. Memperkuat keamanan sistem dengan rutin melakukan pengujian penetrasi, menerapkan autentikasi multi-faktor, enkripsi data yang lebih kuat, serta monitoring keamanan berkelanjutan untuk menghadapi potensi ancaman baru.
4. Mengintegrasikan pembatasan jam operasi harian dengan *firmware* agar perangkat dapat membatasi penggunaan secara mandiri saat terjadi gangguan jaringan, sehingga kontrol waktu tetap berjalan paralel dengan *backend*.
5. Melakukan pengujian performa backend untuk mengevaluasi kemampuan sistem dalam menangani beban akses dari banyak pengguna dan perangkat secara bersamaan, sehingga potensi *bottleneck* dapat diidentifikasi dan dioptimalkan.

DAFTAR PUSTAKA

[1] BPS, "Produksi Tanaman Perkebunan (Ribu Ton) 2023," BPS - Badan Pusat Statistik. Accessed: Nov. 26, 2024. [Online]. Available: https://www.bps.go.id/id/statistics-table/2/MTMyIzI=/produksi-tanaman-perkebunan--ribu-ton-.html

[2] S. H. Luke *et al.*, "Managing Oil Palm Plantations More Sustainably: Large-Scale Experiments Within the Biodiversity and Ecosystem Function in Tropical Agriculture (BEFTA) Programme," *Frontiers in Forests and Global Change*, vol. 2, p. 75, Jan. 2020, doi: 10.3389/ffgc.2019.00075.

[3] N. Abdullah *et al.*, "Towards Smart Agriculture Monitoring Using Fuzzy Systems," *IEEE Access*, vol. 9, pp. 4097–4111, 2021, doi: 10.1109/ACCESS.2020.3041597.

[4] G. Coulby, A. K. Clear, O. Jones, and A. Godfrey, "Low-cost, multimodal environmental monitoring based on the Internet of Things," *Build Environ*, vol. 203, no. 108014, p. 108014, Oct. 2021, doi: 10.1016/j.buildenv.2021.108014.

[5] S. Anweiler and D. Piwowarski, "Multicopter platform prototype for environmental monitoring," *J Clean Prod*, vol. 155, pp. 204–211, Jul. 2017, doi: 10.1016/j.jclepro.2016.10.132.

[6] N. A. M. Pauzi, S. M. Mustaza, and I. Yahya, "Low-cost environmental monitoring mini rover based on IoT technology," *International Journal of Advanced Technology and Engineering Exploration*, vol. 8, no. 74, pp. 64–72, Jan. 2021, doi: 10.19101/IJATEE.2020.S3762190.

[7] M. Rakhra, R. Singh, T. K. Lohani, and M. Shabaz, "Metaheuristic and Machine Learning-Based Smart Engine for Renting and Sharing of Agriculture Equipment," *Math Probl Eng*, vol. 2021, no. 1, pp. 1–13, Feb. 2021, doi: 10.1155/2021/5561065.

[8] I. S. Areni, A. Waridi, I. Amirullah, C. Yohannes, A. Lawi, and A. Bustamin, "IoT-Based of Automatic Electrical Appliance for Smart Home," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 14, no. 18, p. 204, Nov. 2020, doi: 10.3991/ijim.v14i18.15649.

[9] E. Navarro, N. Costa, and A. Pereira, “A Systematic Review of IoT Solutions for Smart Farming,” *Sensors*, vol. 20, no. 15, p. 4231, Jul. 2020, doi: 10.3390/s20154231.

[10] X. Shi *et al.*, “State-of-the-Art Internet of Things in Protected Agriculture,” *Sensors*, vol. 19, no. 8, p. 1833, Apr. 2019, doi: 10.3390/s19081833.

[11] N. A. N. Mohd Adib and S. Daliman, “Conceptual framework of smart fertilization management for oil palm tree based on IOT and deep learning,” *IOP Conf Ser Earth Environ Sci*, vol. 842, no. 1, p. 012072, Aug. 2021, doi: 10.1088/1755-1315/842/1/012072.

[12] J. Krishna Kant, M. Sripaad, A. Bharadwaj, and S. Sundaram, “An Autonomous Hybrid Drone-Rover Vehicle for Weed Removal and Spraying Applications in Agriculture,” *arXiv e-prints*, p. arXiv-2308, 2023, doi: 10.48550/arXiv.2308.04794.

[13] P. Linna, T. Aaltonen, A. Halla, J. Gronman, and N. Narra, “Conceptual design of an autonomous rover with ground penetrating radar: application in characterizing soils using deep learning,” in *Proc. 43rd Int. Conv. Inf., Commun. Electron. Technol. (MIPRO)*, Opatija, Croatia, Sep. 2020, pp. 1134–1139. doi: 10.23919/MIPRO48935.2020.9245270.

[14] B. Das *et al.*, “Designing and development of agricultural rovers for vegetable harvesting and soil analysis,” *PLoS One*, vol. 19, no. 6, p. e0304657, Jun. 2024, doi: 10.1371/journal.pone.0304657.

[15] GeeksforGeeks, “Backend Development,” GeeksforGeeks. Accessed: Dec. 02, 2024. [Online]. Available: https://www.geeksforgeeks.org/backend-development/

[16] D. Inupakutika, G. Rodriguez, D. Akopian, P. Lama, P. Chalela, and A. G. Ramirez, “On the Performance of Cloud-Based mHealth Applications: A Methodology on Measuring Service Response Time and a Case Study,” *IEEE Access*, vol. 10, pp. 53208–53224, 2022, doi: 10.1109/ACCESS.2022.3174855.

[17] M. Albahar, D. Alansari, and A. Jurcut, “An Empirical Comparison of Pen-Testing Tools for Detecting Web App Vulnerabilities,” *Electronics (Basel)*, vol. 11, no. 19, p. 2991, Sep. 2022, doi: 10.3390/electronics11192991.

[18] C. Sravani, P. Kumar, S. Priya, S. K. Yadav, M. J. Rao, and U. D. Prasan, “Constructing a Study Buddy Using MERN (MongoDB, Express.js, React, Node.js) Stack Technologies,” in *IPDIMS 2023*, Basel Switzerland: MDPI, Jul. 2024, p. 27. doi: 10.3390/engproc2024066027.

[19] M. Bawane, “A Review on Technologies used in MERN stack,” *Int J Res Appl Sci Eng Technol*, vol. 10, no. 1, pp. 479–488, Jan. 2022, doi: 10.22214/ijraset.2022.39868.

[20] E. B. Sanjuan, I. A. Cardiel, J. A. Cerrada, and C. Cerrada, “Message Queuing Telemetry Transport (MQTT) Security: A Cryptographic Smart Card Approach,” *IEEE Access*, vol. 8, pp. 115051–115062, 2020, doi: 10.1109/ACCESS.2020.3003998.

[21] A. Makris, K. Tserpes, G. Spiliopoulos, D. Zissis, and D. Anagnostopoulos, “MongoDB Vs PostgreSQL: A comparative study on performance aspects,” *Geoinformatica*, vol. 25, no. 2, pp. 243–268, Apr. 2021, doi: 10.1007/s10707-020-00407-w.

[22] GeeksforGeeks, “Introduction to Postman for API Development,” GeeksforGeeks. Accessed: Dec. 03, 2024. [Online]. Available: https://www.geeksforgeeks.org/introduction-postman-api-development/

[23] Y. Wang, M. V. Mäntylä, Z. Liu, J. Markkula, and P. Raulamo-jurvanen, “Improving test automation maturity: A multivocal literature review,” *Software Testing, Verification and Reliability*, vol. 32, no. 3, p. e1804, May 2022, doi: 10.1002/stvr.1804.

[24] L. F. Da Costa, *Testing JavaScript Applications*. Simon and Schuster, 2021.

[25] Y. Sheffer, D. Hardt, and M. Jones, “JSON Web Token Best Current Practices,” Feb. 2020. doi: 10.17487/RFC8725.

[26] C. Gea, K. J. D. Lase, and M. Syamsudin, “Implementasi Virtual Private Server untuk Mini Hosting,” *JURNAL SAINS DAN KOMPUTER*, vol. 7, no. 01, pp. 5–9, Jan. 2023, doi: 10.61179/jurnalinfact.v7i01.402.

[27] M. Wessel, J. Vargovich, M. A. Gerosa, and C. Treude, “GitHub Actions: The Impact on the Pull Request Process,” *Empir Softw Eng*, vol. 28, no. 6, p. 131, Nov. 2023, doi: 10.1007/s10664-023-10369-w.

[28] R. Setiawan, “Apa Itu CI/CD?,” Dicoding Indonesia. Accessed: Dec. 10, 2024. [Online]. Available: https://www.dicoding.com/blog/apa-itu-ci-cd/

[29] B. George and L. Williams, “A structured experiment of test-driven development,” *Inf Softw Technol*, vol. 46, no. 5, pp. 337–342, Apr. 2004, doi: 10.1016/j.infsof.2003.09.011.

[30] D. Fucci, H. Erdogmus, B. Turhan, M. Oivo, and N. Juristo, “A Dissection of the Test-Driven Development Process: Does It Really Matter to Test-First or to Test-Last?,” *IEEE Transactions on Software Engineering*, vol. 43, no. 7, pp. 597–614, Jul. 2017, doi: 10.1109/TSE.2016.2616877.

[31] D. Janzen and H. Saiedian, “Does Test-Driven Development Really Improve Software Design Quality?,” *IEEE Softw*, vol. 25, no. 2, pp. 77–84, Mar. 2008, doi: 10.1109/MS.2008.34.

[32] N. Amran, F. D. Saiful Bahry, A. H. Abdull Razak, M. S. Muksin, and N. A. Abdul Rahaman, “Database Conceptual Diagram Using ERD and Data

Dictionary Metadata for SME Construction Business," *International Journal of Academic Research in Business and Social Sciences*, vol. 12, no. 10, Oct. 2022, doi: 10.6007/IJARBSS/v12-i10/15142.

[33] Nurbojatmiko, A. Lathifah, F. Bil Amri, and A. Rosidah, "Security Vulnerability Analysis of the Sharia Crowdfunding Website Using OWASP-ZAP," in *2022 10th International Conference on Cyber and IT Service Management (CITSM)*, IEEE, Sep. 2022, pp. 1–5. doi: 10.1109/CITSM56380.2022.9935837.

[34] T. E. Al-jarakh, O. A. Hussein, A. K. Al-azzawi, and M. F. Mosleh, "Design and implementation of IoT based environment pollution monitoring system: A case study of Iraq," *IOP Conf Ser Mater Sci Eng*, vol. 1105, no. 1, p. 012037, Jun. 2021, doi: 10.1088/1757-899X/1105/1/012037.

[35] T. Wahyuni, I. Indriyanti, E. Ermawati, H. Fatah, and N. Ichsan, "Rancang Bangun Sistem Penyewaan Baju Dan Dekorasi Berbasis Web Pada Nita Wedding Organizer," *Jurnal Interkom: Jurnal Publikasi Ilmiah Bidang Teknologi Informasi dan Komunikasi*, vol. 16, no. 1, pp. 1–9, Apr. 2021, doi: 10.35969/interkom.v16i1.91.

[36] S.-Y. Wang, Y.-J. Hsu, S.-J. Hsiao, and W.-T. Sung, "A Sensor Network Web Platform Based on WoT Technology," *Computer Systems Science and Engineering*, vol. 38, no. 2, pp. 197–214, 2021, doi: 10.32604/csse.2021.015713.

[37] A. Apriansyah, "Palembang City Web-Based Futsal Field Rental System Using Express JS and React JS," *Jurnal Sistem dan Teknologi Informasi (JustIN)*, vol. 12, no. 1, p. 16, Jan. 2024, doi: 10.26418/justin.v12i1.68452.

[38] A. Turnip, F. R. Pebriansyah, T. Simarmata, P. Sihombing, and E. Joelianto, "Design of smart farming communication and web interface using MQTT and Node.js," *Open Agric*, vol. 8, no. 1, p. 20220159, Oct. 2023, doi: 10.1515/opag-2022-0159.

[39] ZAP, "Remote OS Command Injection," ZAP by Checkmarx. Accessed: Jul. 25, 2025. [Online]. Available: https://www.zaproxy.org/docs/alerts/90020/

[40] B. Jogi, "CVE-2021-44228: Apache Log4j2 Zero-Day Exploited in the Wild (Log4Shell)," Qualys. Accessed: May 26, 2025. [Online]. Available: https://blog.qualys.com/vulnerabilities-threat-research/2021/12/10/apache-log4j2-zero-day-exploited-in-the-wild-log4shell

[41] ZAP, "Active Scan Rules," ZAP by Checkmarx. Accessed: May 26, 2025. [Online]. Available: https://www.zaproxy.org/docs/desktop/addons/active-scan-rules/#id-40012

[42] ZAP, "User Agent Fuzzer," ZAP by Checkmarx. Accessed: May 18, 2025. [Online]. Available: https://www.zaproxy.org/docs/alerts/10104/