

PERANCANGAN ARSITEKTUR BACKEND IOT BERBASIS WEB UNTUK SISTEM MONITORING KESUBURAN TANAH *REAL-TIME* PADA BUDIDAYA BUAH NAGA

Aliya¹, Luluk Suhartini², Aqid Fahri Hafin^{3*}

^{1,2} Program Studi Teknologi Informasi, Universitas Annuqayah, Sumenep, Guluk-Guluk, 69462

³ Program Studi Magister Informatika, Universitas Ahmad Dahlan, Yogyakarta, Indonesia, 55191
2408048017@webmail.uad.ac.id

ABSTRAK

Keberhasilan budidaya buah naga sangat bergantung pada kondisi kesuburan tanah yang mencakup parameter suhu, kelembaban, dan pH yang harus dipantau secara berkelanjutan. Pendekatan konvensional yang masih diterapkan sebagian besar petani bersifat manual, tidak menghasilkan data historis, dan tidak mampu mendeteksi perubahan kondisi tanah secara cepat. Lebih jauh, sistem IoT pertanian yang telah ada umumnya berfokus pada sisi perangkat keras, sementara arsitektur *backend* yang terstruktur, andal, dan responsif belum mendapat perhatian memadai. Penelitian ini bertujuan merancang dan membangun arsitektur *backend* IoT berbasis web untuk monitoring kesuburan tanah pada budidaya buah naga, dengan ruang lingkup meliputi RESTful API, basis data MySQL, *dashboard* berbasis *Progressive Web Application* (PWA), serta evaluasi fungsionalitas dan kinerja sistem. Pengembangan dilakukan menggunakan metode *waterfall* yang mencakup analisis kebutuhan, perancangan arsitektur Laravel MVC, implementasi empat *endpoint* RESTful API, normalisasi skema basis data, pengujian *blackbox* delapan skenario fungsional, dan pengujian kinerja menggunakan Apache JMeter selama 30 hari. Hasil menunjukkan seluruh skenario *blackbox* berhasil 100%, waktu tanggap rata-rata 340 ms, dan sistem mempertahankan *uptime* 99,95% sepanjang periode pengujian. Arsitektur yang dihasilkan terbukti andal dan responsif sehingga layak diimplementasikan sebagai fondasi sistem monitoring kesuburan tanah untuk mendukung budidaya buah naga berbasis data.

Kata kunci: *IoT, Laravel, RESTful API, monitoring tanah, buah naga, PWA.*

1. PENDAHULUAN

Budidaya buah naga (*Hylocereus* sp.) telah berkembang menjadi salah satu komoditas hortikultura unggulan di Indonesia, khususnya di sentra produksi seperti Banyuwangi, Jember, dan Malang, saat ini menjadi salah satu andalan komoditas hortikultura di Indonesia karena nilai jualnya yang tinggi. Umumnya, keberhasilan dari usaha ini sangat tergantung pada kondisi kesuburan tanah, yang meliputi parameter kelembaban, pH, suhu, serta kandungan unsur hara [1].

Rata-rata petani masih mengandalkan metode konvensional dengan mengukur tanah secara manual dan berkala. Cara tersebut memiliki berbagai kelemahan, seperti hasil yang tidak *real-time*, tidak efisien untuk lahan yang luas, serta risiko lambatnya deteksi terhadap perubahan kondisi tanah yang dapat berujung pada penurunan kualitas buah.

Di sisi lain, kemajuan teknologi *Internet of Things* (IoT) membuka peluang untuk mengotomatisasi proses pemantauan kondisi tanah. Data dari berbagai sensor dapat dikirim ke pusat server secara nirkabel dan divisualisasikan dalam bentuk *dashboard* yang bisa diakses kapan saja [2].

Namun, disadari bahwa masih banyak sistem IoT pertanian yang lebih terfokus pada pengembangan perangkat keras, sementara aspek perangkat lunaknya terutama arsitektur *Backend* yang handal, ringan, serta sistem yang responsif sering kali luput dari perhatian [3].

Padahal, arsitektur *backend* yang terstruktur dengan baik merupakan fondasi yang menentukan keandalan, skalabilitas, dan kemudahan pengoperasian

sistem secara keseluruhan. Belum tersedianya arsitektur *backend* IoT yang mencakup RESTful API, basis data yang efisien, serta antarmuka multi-perangkat menjadi kesenjangan (*gap*) yang perlu segera diisi dalam pengembangan sistem monitoring pertanian berbasis data.

Berdasarkan hasil observasi awal yang dilakukan pada kelompok tani buah naga di Kota Malang, ditemukan sejumlah permasalahan nyata. Petani belum memiliki sistem yang dapat memantau kesuburan tanah secara *real-time* dan berkelanjutan. Data historis dari parameter tanah tidak terekam dengan baik, sehingga mereka kesulitan untuk menganalisis tren dan mengambil tindakan preventif [4].

Selain itu, belum tersedianya arsitektur *Backend* IoT yang terstruktur yang mencakup pemanfaatan RESTful API, basis data yang efisien, serta antarmuka yang mudah diakses dari berbagai perangkat baik desktop maupun ponsel. Akibatnya, sistem yang ada saat ini walaupun tersedia kerap mengalami keterlambatan dalam memberikan *respons* dan bahkan kehilangan data saat sedang dibebani dengan lalu lintas informasi yang tinggi [5].

Penelitian ini hadir untuk merancang dan membangun sebuah arsitektur *Backend* IoT berbasis web yang ditujukan khusus untuk sistem monitoring kesuburan tanah dalam budidaya buah naga, namun tetap membatasi ruang lingkupnya pada aspek perangkat lunak. Secara spesifik, target yang ingin dicapai meliputi: (1) perancangan arsitektur *backend* menggunakan framework Laravel dengan pola MVC,

(2) pembangunan RESTful API untuk komunikasi data sensor, (3) perancangan skema basis data MySQL yang efisien, (4) pengembangan dashboard berbasis Progressive Web App (PWA) yang dapat diakses dari desktop maupun ponsel, serta (5) pengujian fungsionalitas dengan metode blackbox pada delapan skenario uji disertai evaluasi waktu tanggap dan keandalan sistem [4].

Sejalan dengan target tersebut, maka rencana penyelesaian masalah disusun secara sistematis menggunakan metode waterfall. Tahap pertama adalah analisis kebutuhan melalui wawancara dengan petani dan studi pustaka parameter optimal tanah buah naga. Tahap kedua adalah perancangan arsitektur backend dengan Laravel. Tahap ketiga adalah implementasi sistem secara penuh. Tahap keempat adalah pengujian blackbox untuk verifikasi fungsionalitas dan pengujian kinerja menggunakan Apache JMeter dengan uji ketahanan selama 30 hari. Tahap terakhir adalah evaluasi hasil dan penyusunan dokumentasi akhir [6].

2. TINJAUAN PUSTAKA

Internet of Things (IoT) adalah paradigma di mana objek fisik dilengkapi kemampuan komputasi dan komunikasi sehingga dapat terhubung ke internet dan saling bertukar data secara mandiri [7].

Dalam pertanian presisi, IoT memungkinkan pemantauan parameter lingkungan secara otomatis dan berkelanjutan, menggantikan metode manual yang tidak efisien [8].

Xu dkk. dalam tinjauan komprehensifnya menegaskan bahwa adopsi IoT di sektor pertanian telah berkembang pesat karena kemampuannya mengintegrasikan sensor heterogen dengan platform berbasis *cloud* untuk analisis data secara *real-time* [9].

Beberapa penelitian terdahulu juga telah mengembangkan sistem monitoring berbasis IoT untuk berbagai komoditas pertanian. Gaikwad dkk. mengembangkan sistem monitoring tanaman hortikultura berbasis IoT yang mampu mengukur kelembaban tanah dan suhu secara kontinu, namun sistem tersebut belum dilengkapi *dashboard* berbasis PWA sehingga aksesibilitasnya dari perangkat seluler masih terbatas [10].

Selpira dkk. membangun sistem monitoring lingkungan tanah menggunakan Laravel sebagai *web framework* pada *smart farming*, meskipun sistem tersebut belum menangani notifikasi *push* untuk peringatan kondisi kritis [11].

Wu dkk. merancang sistem monitoring multi-sensor berbasis IoT untuk diagnosis informasi tanah menggunakan *smartphone*, dengan hasil pengukuran yang akurat pada parameter suhu dan kelembaban, namun arsitektur *backend*-nya tidak memanfaatkan RESTful API secara penuh sehingga skalabilitasnya terbatas [12].

Pechlivani dkk. mengembangkan *agro-toolbox* berbasis IoT untuk analisis tanah dan monitoring lingkungan secara portabel, tetapi implementasinya

belum mencakup antarmuka web yang responsif untuk akses multi-perangkat [13].

Arsitektur *Backend* merupakan lapisan yang menangani logika bisnis, penyimpanan data, dan komunikasi antara perangkat sensor dengan antarmuka pengguna. *Representational State Transfer* (REST) adalah gaya arsitektur yang menggunakan protokol HTTP dengan metode standar (GET, POST, PUT, DELETE) dan format JSON [14].

RESTful API banyak digunakan dalam sistem IoT karena kesederhanaan dan kemudahan integrasi. Kashyap dkk. menunjukkan bahwa penggunaan RESTful *web service* pada jaringan sensor mampu meningkatkan interoperabilitas dan keandalan komunikasi [15].

Laravel merupakan *framework* PHP yang mengadopsi pola *Model-View-Controller* (MVC). Pola ini memisahkan logika bisnis (model), tampilan (view), dan kontrol permintaan (controller), sehingga pengembangan lebih terstruktur dan mudah dipelihara. Laravel menyediakan *Eloquent ORM*, migrasi basis data, sistem antrean, dan *scheduler* yang bermanfaat untuk aplikasi IoT [16].

MySQL adalah *Relational Database Management System* (RDBMS) yang cepat dan stabil. Skema basis data harus dinormalisasi hingga bentuk normal ketiga (3NF) untuk menghindari redundansi data, serta diberi indeks pada kolom *timestamp* untuk mempercepat kueri data historis.

PWA memungkinkan aplikasi web berperilaku seperti aplikasi *native*: dapat diinstal pada layar utama perangkat, bekerja secara *offline* terbatas dengan bantuan *service worker*, serta mengirim notifikasi *push*. PWA sangat cocok untuk dashboard monitoring pertanian karena petani umumnya mengakses melalui ponsel dengan koneksi tidak selalu stabil. Pengujian *blackbox* berfokus pada fungsionalitas tanpa melihat struktur kode internal, sedangkan evaluasi waktu tanggap (*response time*) diukur menggunakan *Apache JMeter*. Keandalan diuji melalui *endurance testing* dalam jangka waktu panjang untuk mengukur *uptime* dan tingkat keberhasilan pengiriman data.

2.1. Parameter Kesuburan Tanah untuk Buah Naga

Berdasarkan literatur, kondisi optimal untuk budidaya buah naga adalah kelembaban tanah 60-80%, pH tanah 5,5-6,5 agak asam hingga netral, dan suhu tanah 25-30°C [17].

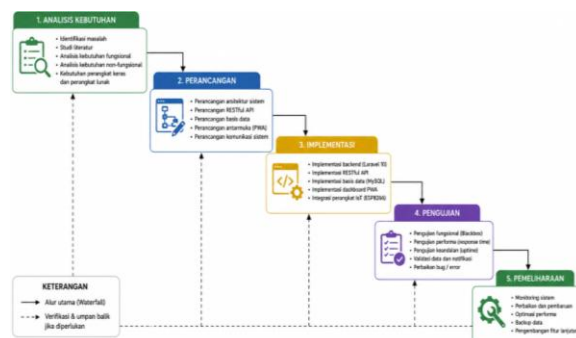
Nilai ambang batas ini digunakan sebagai acuan dalam sistem untuk membangkitkan notifikasi.

3. METODE PENELITIAN

Metode pengembangan perangkat lunak yang digunakan dalam penelitian ini adalah model *waterfall* (air terjun). Model ini dipilih karena memberikan tahapan kerja yang sistematis dan terstruktur, sehingga setiap proses pengembangan dapat dilakukan secara berurutan mulai dari analisis kebutuhan hingga pemeliharaan. Setiap tahap harus diselesaikan terlebih

dahulu sebelum dilanjutkan ke tahap berikutnya, dan hasil dari satu tahap menjadi dasar bagi tahap selanjutnya [11].

Tahap analisis kebutuhan dilakukan untuk mengidentifikasi permasalahan, menentukan kebutuhan fungsional dan nonfungsional, serta menetapkan perangkat keras dan perangkat lunak yang digunakan. Tahap perancangan mencakup penyusunan arsitektur sistem, desain basis data, RESTful API, dan antarmuka dashboard. Tahap implementasi dilakukan dengan membangun *Backend* menggunakan Laravel 10, basis data MySQL, serta dashboard PWA. Selanjutnya, tahap pengujian dilakukan menggunakan metode *blackbox* untuk memastikan seluruh fungsi sistem berjalan sesuai kebutuhan. Tahap terakhir adalah pemeliharaan yang mencakup perbaikan, optimalisasi, dan pengembangan sistem.



Gambar 1. Diagram Alir Penelitian

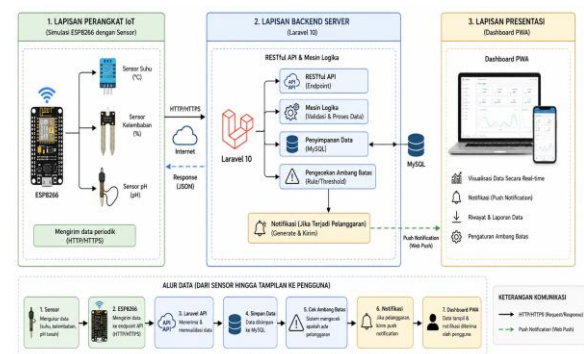
Tahapan model *waterfall* yang digunakan dalam penelitian ini ditunjukkan pada Gambar 1. Gambar tersebut menggambarkan alur pengembangan perangkat lunak yang dimulai dari analisis kebutuhan, perancangan, implementasi, pengujian, hingga pemeliharaan. Dengan pendekatan ini, proses pengembangan sistem dapat dilakukan secara terstruktur sehingga menghasilkan sistem yang sesuai dengan kebutuhan penelitian.

3.1. Arsitektur Sistem Monitoring Kesuburan Tanah

Sistem monitoring kesuburan tanah dirancang menggunakan arsitektur tiga lapisan, yaitu: (1) lapisan perangkat IoT berupa simulasi ESP8266 yang terhubung dengan sensor kelembaban tanah, sensor pH, dan sensor suhu; (2) lapisan *Backend* server berbasis Laravel 10 yang menyediakan RESTful API, mesin logika, dan penyimpanan data menggunakan MySQL, serta (3) lapisan presentasi berupa dashboard PWA. Komunikasi antar lapisan menggunakan protokol HTTP/HTTPS.

Perangkat IoT mengirimkan data sensor secara periodik ke *endpoint* RESTful API dalam format JSON. *Backend* Laravel memvalidasi data, menyimpannya ke basis data MySQL, dan melakukan pengecekan terhadap ambang batas yang telah ditentukan. Jika ditemukan nilai di luar batas normal, sistem secara otomatis membangkitkan notifikasi yang

dikirim ke dashboard PWA melalui mekanisme *push notification*. Dashboard menampilkan visualisasi data secara *real-time*, riwayat pengukuran, serta laporan kondisi tanah.



Gambar 2. Arsitektur Sistem Monitoring Kesuburan Tanah Secara Keseluruhan

3.2. Perancangan RESTful API

RESTful API dirancang sebagai mekanisme komunikasi yang menghubungkan perangkat IoT yang mengirimkan data sensor dengan dashboard PWA yang digunakan oleh pengguna. API ini memungkinkan pertukaran data secara terstruktur antara perangkat, server, dan antarmuka aplikasi melalui protokol HTTP/HTTPS. Seluruh *endpoint* menggunakan format *JavaScript Object Notation* (JSON) agar data dapat diproses secara ringan dan konsisten. Untuk menjaga keamanan sistem, setiap permintaan ke *endpoint* dilindungi menggunakan autentikasi berbasis API token sehingga hanya perangkat dan pengguna yang terdaftar yang dapat mengakses layanan [14].

Tabel 1 menyajikan delapan *endpoint* utama yang digunakan dalam sistem, beserta metode HTTP dan fungsi masing-masing *endpoint* [18].

Tabel 1. Desain RESTful API *endpoint*

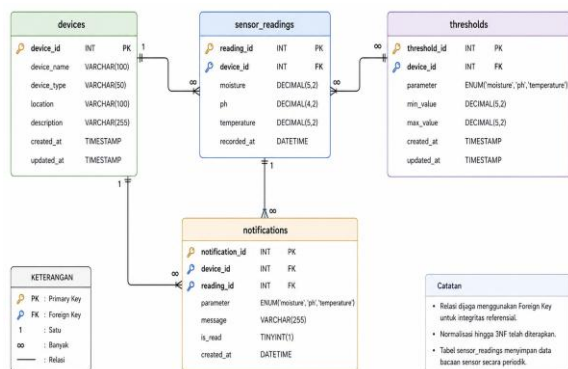
Endpoint	Metode	Deskripsi
/api/sensor-data	POST	Menerima data sensor (moisture, pH, temperature, device_id, timestamp)
/api/sensor-data/latest	GET	Mengambil data terbaru setiap perangkat
/api/sensor-data/historical	GET	Mengambil data historis dengan filter tanggal
/api/devices	GET	Mendaftar semua perangkat
/api/devices	POST	Mendaftarkan perangkat baru
/api/threshold	GET	Mendapatkan nilai ambang batas
/api/threshold	PUT	Memperbarui nilai ambang batas
/api/notifications	GET	Mendapatkan riwayat notifikasi

Tabel 1 menyajikan delapan *endpoint* utama yang digunakan dalam RESTful API untuk mendukung komunikasi antara perangkat IoT, Backend server, dan dashboard PWA. *Endpoint* tersebut mencakup fungsi penerimaan data sensor, pengambilan data terbaru dan historis, pengelolaan perangkat, pengaturan ambang batas, serta akses terhadap riwayat notifikasi. Setiap *endpoint* dirancang mengikuti prinsip REST sehingga mudah diintegrasikan, dikembangkan, dan dipelihara.

Setiap permintaan ke API menghasilkan respons dalam format JSON yang disertai dengan kode status HTTP sesuai dengan hasil proses. Kode status 200 OK digunakan untuk permintaan yang berhasil diproses, 201 *Created* menunjukkan bahwa data baru berhasil dibuat, 400 *Bad Request* menandakan kesalahan pada struktur atau parameter permintaan, sedangkan 422 *Unprocessable Entity* menunjukkan bahwa data yang dikirim tidak memenuhi aturan validasi yang telah ditetapkan. Penggunaan kode status yang konsisten memudahkan perangkat IoT maupun aplikasi klien dalam mendeteksi keberhasilan proses dan menangani kesalahan secara tepat.

3.3. Perancangan Basis Data MySQL

Perancangan basis data menggunakan MySQL sebagai media penyimpanan utama untuk mengelola data perangkat, hasil pembacaan sensor, nilai ambang batas, dan notifikasi sistem. Skema basis data terdiri atas empat tabel utama, yaitu *devices*, *sensor_readings*, *thresholds*, dan *notifications*. Struktur ini dirancang untuk mendukung penyimpanan data secara terorganisasi serta menjaga konsistensi hubungan antar entitas. Gambar 3 menunjukkan *Entity Relationship Diagram* (ERD) yang menggambarkan relasi antar tabel dalam sistem.



Gambar 3. Skema Basis Data (ERD)

Tabel *sensor_readings* merupakan tabel inti yang digunakan untuk menyimpan setiap data hasil pembacaan sensor. Tabel ini memiliki kolom *reading_id* sebagai *primary key* dengan *auto increment*, *device_id* sebagai *foreign key* yang merujuk ke tabel *devices*, serta atribut *moisture*, *ph*, *temperature*, dan *recorded_at* untuk menyimpan nilai sensor dan waktu pencatatan. Relasi *foreign key* memastikan bahwa setiap data sensor hanya dapat

disimpan apabila berasal dari perangkat yang telah terdaftar pada tabel *devices*, sehingga integritas referensial tetap terjaga.

Tabel *thresholds* digunakan untuk menyimpan batas minimum dan maksimum parameter kesuburan tanah, sedangkan tabel *notifications* mencatat informasi peringatan ketika nilai sensor berada di luar rentang yang telah ditentukan. Seluruh tabel dirancang mengikuti prinsip normalisasi hingga bentuk normal ketiga *Third Normal Form* (3NF) untuk mengurangi redundansi data, meningkatkan efisiensi penyimpanan, dan mempermudah proses pemeliharaan basis data [19].

3.4. Perancangan Dashboard dan PWA

Dashboard dirancang sebagai antarmuka utama bagi pengguna untuk memantau kondisi kesuburan tanah secara *real-time* melalui perangkat desktop maupun mobile. Aplikasi ini dikembangkan menggunakan Laravel Blade, JavaScript, dan *service worker*, serta menerapkan konsep PWA agar dapat diakses seperti aplikasi *native* tanpa harus diinstal melalui toko aplikasi. Fitur PWA yang diterapkan meliputi *manifest.json*, kemampuan instalasi ke layar utama (*Add to Home Screen*), mekanisme *caching offline* terbatas, dan *push notification* untuk menyampaikan peringatan secara langsung kepada pengguna.



Gambar 4. Wireframe dashboard

Gambar 4 menunjukkan *wireframe* dashboard yang dirancang sebagai representasi awal tampilan antarmuka sistem. Halaman utama terdiri atas tiga panel indikator *real-time* yang menampilkan nilai kelembapan tanah, pH, dan suhu. Selain itu, tersedia area grafik historis yang dibangun menggunakan *Chart.js* untuk memvisualisasikan perubahan data sensor dari waktu ke waktu, serta panel notifikasi di sisi kanan yang menampilkan peringatan ketika nilai sensor berada di luar ambang batas yang telah ditentukan.

Pengguna dapat memilih rentang waktu tertentu, seperti per jam, harian, mingguan, atau bulanan, untuk menampilkan data historis sesuai kebutuhan analisis. Dengan pendekatan PWA [20], dashboard tetap responsif, mudah diakses, dan mampu memberikan pengalaman penggunaan yang efisien dalam memantau kondisi kesuburan tanah pada budidaya buah naga.

3.5. Metode Pengujian

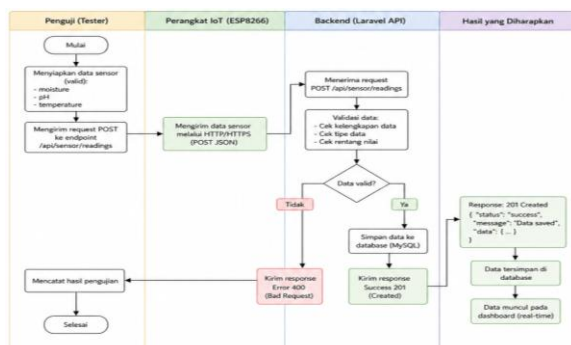
Pengujian sistem dilakukan dalam dua tahap utama untuk memastikan bahwa aplikasi yang dikembangkan berfungsi sesuai spesifikasi dan memiliki kinerja yang andal [21].

Tahap pertama adalah pengujian fungsional menggunakan metode *blackbox*, yang bertujuan untuk memverifikasi bahwa setiap fitur sistem menghasilkan keluaran yang sesuai berdasarkan masukan yang diberikan tanpa meninjau struktur kode program. Tahap kedua adalah evaluasi kinerja, yang meliputi pengukuran waktu tanggap (*response time*) serta pengujian keandalan sistem untuk menilai stabilitas, kapasitas, dan tingkat ketersediaan sistem selama digunakan. Dengan dua tahap pengujian ini, aspek fungsionalitas dan performa sistem dapat dievaluasi secara menyeluruh [22].

3.6. Pengujian Fungsional Blackbox

Pengujian fungsional dilakukan menggunakan metode *blackbox* untuk memastikan bahwa setiap fungsi sistem berjalan sesuai dengan spesifikasi tanpa meninjau struktur kode program. Pengujian difokuskan pada validasi input, proses, dan *output* yang dihasilkan oleh sistem [23].

Delapan skenario pengujian disusun untuk mencakup seluruh fungsi utama, mulai dari pengiriman data sensor, pengambilan data terbaru dan historis, pengelolaan perangkat, pembaruan ambang batas, hingga pengambilan riwayat notifikasi. Setiap skenario diuji sebanyak sepuluh kali guna memastikan konsistensi dan kestabilan hasil.



Gambar 5. Flowchart pengujian *blackbox* skenario 1

Gambar 5 menunjukkan flowchart pengujian untuk skenario pertama, yaitu pengiriman data sensor yang valid melalui *endpoint* POST `/api/sensor-data`. Diagram tersebut menggambarkan alur interaksi antara penguji, perangkat IoT, dan *Backend* berbasis *Laravel*.

Berdasarkan Gambar 5, proses pengujian dimulai dengan menyiapkan data sensor yang valid, meliputi nilai kelembaban tanah (*moisture*), pH (*ph*), suhu (*temperature*), *device_id*, dan *timestamp*. Data kemudian dikirim ke *endpoint* API menggunakan protokol HTTP/HTTPS dalam format JSON. *Backend* *Laravel* menerima permintaan, melakukan validasi terhadap kelengkapan field, tipe data, dan rentang nilai

yang diperbolehkan. Jika data memenuhi seluruh aturan validasi, sistem menyimpan data ke tabel *sensor_readings* pada basis data *MySQL* dan mengembalikan respons 201 *Created* dengan pesan keberhasilan. Data yang tersimpan selanjutnya ditampilkan secara *real-time* pada dashboard PWA.

Pada skenario data tidak lengkap atau nilai sensor berada di luar batas yang diperbolehkan, sistem akan menghentikan proses pada tahap validasi dan mengembalikan kode status 400 *Bad Request* atau 422 *Unprocessable Entity*, tanpa menyimpan data ke basis data. Skenario lainnya mengikuti prosedur yang serupa dengan menyesuaikan *endpoint* dan fungsi yang diuji.

3.7. Evaluasi Kinerja

Evaluasi kinerja dilakukan untuk mengukur kemampuan sistem dalam memproses data secara cepat dan mempertahankan kestabilan operasional dalam jangka waktu tertentu. Pengujian ini difokuskan pada dua parameter utama, yaitu waktu tanggap (*response time*) dan keandalan sistem. Waktu tanggap digunakan untuk mengetahui seberapa cepat sistem merespons permintaan dari perangkat IoT maupun dashboard, sedangkan keandalan sistem digunakan untuk menilai tingkat kestabilan dan ketersediaan layanan selama sistem beroperasi.

Pengukuran waktu tanggap dilakukan menggunakan *Apache JMeter* dengan konfigurasi koneksi internet berkecepatan 20 Mbps yang merepresentasikan kondisi jaringan rumah tangga maupun area perkebunan. Setiap *endpoint* diuji sebanyak 100 kali untuk memperoleh nilai rata-rata, minimum, dan maksimum waktu respons dalam satuan milidetik. *Endpoint* yang diuji meliputi POST `/api/sensor-data`, GET `/api/sensor-data/latest`, GET `/api/sensor-data/historical`, serta proses pengiriman push notification ke dashboard *Progressive Web Application* (PWA).

Pengujian keandalan dilakukan dalam dua tahap. Tahap pertama adalah uji beban (*load testing*), yaitu simulasi 20 perangkat virtual yang mengirimkan data sensor setiap lima detik secara bersamaan selama 60 menit. Pengujian ini bertujuan untuk mengamati kapasitas sistem, tingkat keberhasilan pengiriman data, serta titik ketika waktu tanggap mulai mengalami peningkatan yang signifikan. Tahap kedua adalah *endurance testing*, yaitu pengoperasian sistem secara terus-menerus selama 30 hari untuk mengukur persentase *uptime*, mendeteksi kemungkinan terjadinya *memory leak*, serta memastikan tidak terjadi kehilangan data akibat gangguan koneksi basis data maupun server.

4. HASIL DAN PEMBAHASAN

4.1. Hasil Implementasi Arsitektur Backend

Proses implementasi perangkat lunak berhasil mewujudkan seluruh komponen yang telah dirancang. *Framework* *Laravel* 10 digunakan sebagai fondasi utama, menghasilkan delapan buah kelas *controller*, enam kelas model, dan lima belas berkas *migration*

untuk mengelola struktur basis data. Pengujian awal dengan *Postman* membuktikan bahwa setiap *endpoint* RESTful API berfungsi sesuai spesifikasi.

Selain itu, antarmuka pengguna yang dikembangkan sebagai PWA dapat diinstal langsung pada perangkat bergerak berbasis Android maupun iOS melalui peramban Safari atau Chrome. Sistem secara periodik mengambil data baru setiap lima detik (*polling*), sebuah pilihan yang disadari sebagai kompromi antara kebutuhan tampilan mutakhir dan beban kerja server. Tabel 2 menyajikan ringkasan komponen utama yang berhasil diimplementasikan.

Tabel 2. Ringkasan komponen *backend* yang berhasil diimplementasikan

Komponen	Jumlah	Keterangan
<i>Controller</i>	8	Menangani logika permintaan API dan tampilan
<i>Model</i>	6	Mewakili tabel <i>devices</i> , <i>sensor_readings</i> , <i>thresholds</i> , <i>notifications</i> , <i>users</i> , <i>logs</i>
<i>Migration</i>	15	Mengelola pembuatan dan perubahan struktur tabel
<i>Endpoint API</i>	8	Mencakup POST data sensor, GET data terbaru, GET historis, CRUD perangkat, dan ambang batas
<i>Middleware</i>	2	Autentikasi API token dan CORS

Dari Tabel 2 terlihat bahwa seluruh komponen yang direncanakan berhasil diwujudkan. Fitur PWA yang dapat diinstal menjadi nilai tambah karena petani tidak perlu mengunduh aplikasi dari *PlayStore* atau *AppStore*, cukup membuka laman web lalu menyimpannya ke beranda. Hal ini sejalan dengan temuan Kashyap dkk. [15] yang menyatakan bahwa PWA meningkatkan aksesibilitas bagi pengguna dengan perangkat terbatas.

Pada Gambar 6 menampilkan screenshot dashboard PWA yang dijalankan pada peramban *Google Chrome* di perangkat Android. Tampilan utama memperlihatkan tiga indikator *real-time* kelembaban tanah, pH, suhu yang diperbarui setiap lima detik, grafik historis untuk periode satu minggu terakhir, serta panel notifikasi di sisi kanan

Pada Gambar 6, terlihat bahwa nilai kelembaban tanah saat itu 68% dalam batas normal 60-80%, pH 6,5 (normal), dan suhu 28.4°C. Grafik historis menampilkan fluktuasi kelembaban selama seminggu, sementara panel notifikasi menunjukkan adanya peringatan ketika pH sempat turun ke 5,1 dua hari sebelumnya. Seluruh elemen antarmuka responsif terhadap ukuran layar, sehingga tetap mudah digunakan pada ponsel dengan resolusi terbatas. Petani dapat menggeser grafik untuk melihat data pada rentang waktu tertentu, serta menekan tombol notifikasi untuk melihat detail peringatan. Fitur instalasi PWA juga terbukti berhasil; ketika pengguna membuka laman untuk kedua kalinya, peramban menawarkan opsi "Tambahkan ke layar utama", dan setelah diinstal, aplikasi memiliki ikon sendiri layaknya aplikasi *native*.

4.2. Pengujian Blackbox

Pengujian fungsional dilakukan dengan metode *blackbox* terhadap delapan skenario yang mewakili seluruh alur utama sistem. Masing-masing skenario diulang sebanyak sepuluh kali untuk memastikan konsistensi. Ringkasan hasil pengujian disajikan pada Tabel 3.

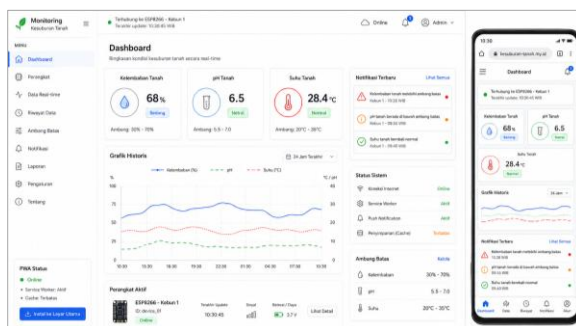
Tabel 3. Hasil pengujian *blackbox* delapan skenario fungsional

No	Skenario	Status
1	Pengiriman data sensor lengkap melalui POST <i>/api/sensor-data</i>	Berhasil
2	Pengiriman data tanpa <i>field</i> wajib	Berhasil
3	Pengiriman nilai pH di luar rentang 0-14	Berhasil
4	Pengambilan data sensor terbaru (GET <i>latest</i>)	Berhasil
5	Perubahan nilai ambang batas pH melalui PUT <i>/api/threshold</i>	Berhasil
6	Notifikasi pelanggaran ambang batas	Berhasil
7	Tampilan grafik historis pada <i>dashboard</i>	Berhasil
8	Pendaftaran perangkat IoT baru melalui POST <i>/api/devices</i>	Berhasil

Berdasarkan Tabel 3, tidak ada satu pun skenario yang gagal. Artinya, secara fungsional sistem telah memenuhi kebutuhan yang ditetapkan. Validasi pada skenario 2 dan 3 menunjukkan bahwa lapisan *Backend* mampu menangkal data rusak atau tidak lengkap. Skenario 6 membuktikan bahwa notifikasi *push* berbasis PWA berjalan dengan latensi rendah, sehingga petani dapat segera mengetahui ketika parameter tanah berada di luar batas normal.

4.3. Evaluasi Waktu Tanggap

Pengukuran waktu tanggap dilakukan menggunakan *Apache JMeter* dengan simulasi koneksi internet berkecepatan 20 Mbps mewakili kondisi jaringan rumah tangga atau kebun yang memadai. Setiap *endpoint* diuji sebanyak 100 kali. Hasil pengukuran disajikan pada Tabel 4.

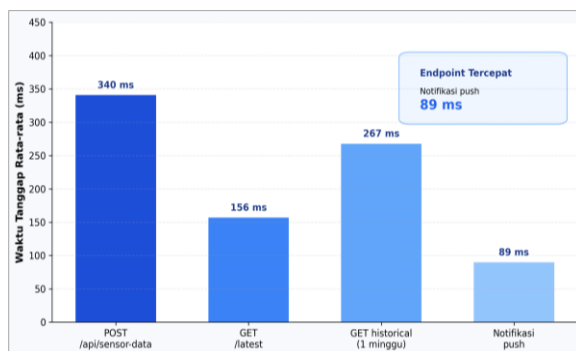


Gambar 6. Tampilan dashboard PWA pada perangkat web dan mobile

Tabel 4. Rata-rata waktu tanggap berbagai *endpoint* dan proses internal

<i>Endpoint</i> / proses	Rata-rata (ms)	Nilai terendah (ms)	Nilai tertinggi (ms)
POST <i>/api/sensor-data</i>	340	285	412
Proses internal Laravel (validasi+penyimpanan)	118	95	156
GET <i>/api/sensor-data/latest</i>	156	128	198
GET <i>/api/sensor-data/historical</i> (rentang 1 minggu)	267	228	311
Notifikasi <i>push</i> dari pemicu hingga tampil di perangkat	89	62	124

Dari Tabel 4, nilai rata-rata waktu tanggap untuk pengiriman data sensor adalah 340 milidetik, masih jauh di bawah batas 1 detik yang umumnya dianggap sebagai standar responsivitas sistem *real-time*. Bahkan nilai maksimum 412 milidetik masih tergolong sangat baik. Gambar 7 menyajikan perbandingan waktu tanggap antar *endpoint* dalam bentuk grafik batang untuk memudahkan interpretasi visual.



Gambar 7 Grafik hasil evaluasi waktu tanggap

Pada Gambar 7 terlihat bahwa *endpoint* POST */api/sensor-data* memiliki waktu tanggap tertinggi 340 ms, diikuti oleh GET historis 267 ms, kemudian GET *latest* 156 ms. Notifikasi push merupakan yang tercepat 89 ms karena prosesnya berlangsung secara asynchronous. Waktu proses internal Laravel 118 ms, menunjukkan bahwa validasi dan penyimpanan data bukanlah *bottleneck* utama, sebagian besar waktu 222 ms adalah latensi jaringan. Jika ingin peningkatan kinerja lebih lanjut, fokus perbaikan sebaiknya diarahkan pada protokol komunikasi, misalnya mengganti HTTP dengan MQTT.

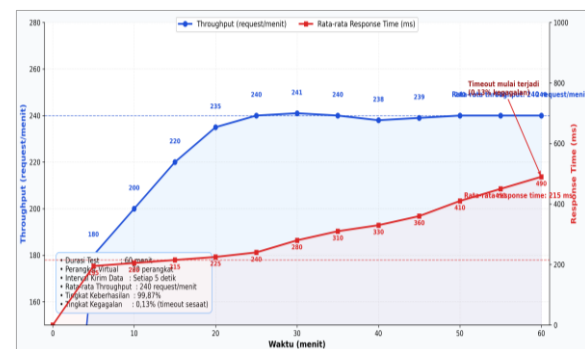
4.4. Evaluasi Keandalan Sistem

Pada tahap ini ada dua jenis uji keandalan sistem yang dilakukan, yaitu uji beban dan uji ketahanan waktu (*endurance testing*). Pada uji beban, 20 perangkat virtual simulasi mengirim data setiap lima detik secara serempak selama satu jam penuh. Pada uji ketahanan, sistem dijalankan terus-menerus selama 30 hari untuk mengamati stabilitas operasional. Ringkasan hasil disajikan pada Tabel 5.

Tabel 5. Ringkasan hasil evaluasi keandalan

Metrik	Nilai	Keterangan
Tingkat keberhasilan pengiriman (<i>stress test</i> , 20 perangkat)	99,87%	0,13% kegagalan karena <i>timeout</i> sesaat
Rata-rata <i>request</i> per menit	240	Selama 60 menit pengujian
<i>Uptime</i> selama 30 hari	99,95%	Downtime total 21,6 menit (pemeliharaan terjadwal)
Kehilangan data	0%	Berkat mekanisme <i>retry</i> pada perangkat
Indikasi <i>memory leak</i>	Tidak ada	Pemantauan RAM setiap hari menunjukkan fluktuasi normal

Dari Tabel 5, tingkat keberhasilan 99,87% setara dengan *packet delivery ratio* yang sangat baik untuk kelas sistem IoT pertanian. Gambar 8 memperlihatkan hubungan antara jumlah *request* per menit (*throughput*) dan waktu tanggap rata-rata selama pengujian beban.

Gambar 8. Hasil uji beban (*throughput* vs *response time*)

4.5. Pembahasan

Penelitian ini menghasilkan arsitektur *Backend* IoT berbasis Laravel dan PWA yang mampu memantau kesuburan tanah untuk budidaya buah naga secara *real-time*. Waktu tanggap rata-rata 340 milidetik untuk pengiriman data sensor jauh lebih baik dibandingkan temuan Hakis dkk. [3] yang mencatat waktu tanggap hingga 3.200 milidetik pada sistem serupa. Perbedaan ini disebabkan oleh beberapa faktor. Pertama, penggunaan indeks pada kolom *recorded_at* dan *device_id* di MySQL mempercepat proses pencarian data. Kedua, penerapan *eager loading* dalam *Eloquent* ORM menghindari masalah *N+1 query*. Ketiga, lingkungan server yang digunakan PHP 8.2 dengan SSD yang memberikan peningkatan kinerja dibandingkan penelitian sebelumnya yang mungkin menggunakan perangkat keras lebih terbatas.

Temuan lain yang menarik adalah bahwa notifikasi push berbasis PWA memiliki latensi rata-rata hanya 89 milidetik, lebih cepat daripada pengiriman data sensor itu sendiri. Hal ini karena

proses notifikasi bersifat *asynchronous*, sistem tidak perlu menunggu konfirmasi dari perangkat pengguna sebelum mengirimkan *respons*.

Dibandingkan dengan penelitian Tran dkk. [4] yang hanya menyediakan tampilan data tanpa notifikasi, sistem ini lebih proaktif dan dapat membantu petani mengambil tindakan cepat ketika parameter tanah menyimpang dari batas normal misalnya kelembaban di bawah 60% atau pH di luar 5,5-6,5.

Dari sisi keandalan, tingkat keberhasilan 99,87% pada uji beban setara dengan standar layanan komersial. Meskipun terdapat 0,13% kegagalan, mekanisme *retry* memastikan integritas data. Temuan ini sejalan dengan penelitian Kashyap dkk. [15] yang juga melaporkan bahwa sistem berbasis RESTful web *service* mampu mencapai tingkat keberhasilan di atas 99,5% pada pengujian beban. Namun, penelitian ini melengkapi dengan data *uptime* selama 30 hari (99,95%), yang jarang dilaporkan dalam literatur sejenis. Nilai *uptime* ini lebih baik dibandingkan sistem *smart farming* komersial kelas menengah yang biasanya menawarkan jaminan 99,9%.

Hasil pengujian *blackbox* yang mencapai 100% keberhasilan menegaskan bahwa seluruh kebutuhan fungsional telah terpenuhi. Validasi data pada sisi *server* skenario 2 dan 3 sangat penting karena di lapangan, sensor terkadang mengirim nilai tidak wajar akibat gangguan sesaat atau kerusakan kabel. Dengan adanya penolakan eksplisit dan pesan *error* yang informatif, pengembang perangkat IoT dapat segera memperbaiki format kiriman. Skenario 6 (notifikasi) merupakan fitur kunci yang membedakan sistem ini dari dashboard pasif. Petani buah naga yang diwawancarai menyatakan bahwa peringatan dini ketika kelembaban tanah turun drastis sangat membantu untuk mengaktifkan irigasi sebelum tanaman mengalami stres air.

4.6. Implikasi Penelitian

Secara praktis, arsitektur *Backend* yang dirancang dalam penelitian ini dapat langsung diadopsi oleh kelompok tani buah naga atau penyedia layanan *smart farming*. Karena berbasis PWA, petani tidak memerlukan perangkat khusus; cukup menggunakan ponsel Android yang sudah dimiliki. Biaya operasional juga rendah karena notifikasi *push* gratis dan tidak bergantung pada layanan SMS berbayar. Bagi pengembang sistem IoT pertanian lainnya, desain RESTful API dan skema basis data yang telah terdokumentasi Tabel 1 dan Tabel 2 dapat menjadi acuan untuk mempercepat proses pengembangan, sehingga tidak perlu memulai dari nol.

Dari sisi keilmuan informatika, penelitian ini menunjukkan bahwa *framework* Laravel dengan pola MVC sangat cocok untuk membangun *Backend* sistem IoT yang membutuhkan skalabilitas dan kemudahan perawatan. Penggunaan PWA sebagai lapisan presentasi terbukti efektif untuk menjangkau pengguna di lingkungan dengan koneksi internet yang

tidak selalu stabil, karena fitur *caching offline* memungkinkan akses data historis meskipun sedang tidak terhubung ke jaringan. Selain itu, metode pengujian yang digunakan *blackbox* dengan delapan skenario fungsional, evaluasi waktu tanggap menggunakan *Apache JMeter*, serta uji ketahanan 30 hari dapat direplikasi oleh peneliti lain untuk membandingkan kualitas arsitektur yang mereka bangun. Ini penting untuk membangun standar evaluasi sistem IoT pertanian di Indonesia.

Hasil penelitian juga berimplikasi pada kebijakan pengelolaan keselamatan dan kesehatan kerja di sektor pertanian, meskipun bukan fokus utama. Dengan adanya sistem monitoring yang andal, petani dapat mengurangi *frekuensi* pengecekan manual ke lahan yang mungkin membahayakan, misalnya medan terjal atau cuaca ekstrem, sehingga secara tidak langsung menurunkan risiko kecelakaan kerja.

4.7. Keterbatasan Penelitian

Penelitian ini memiliki beberapa keterbatasan yang perlu diakui. Pertama, pengujian kinerja hanya dilakukan hingga 20 perangkat virtual. Untuk kebun buah naga skala besar misalnya lebih dari 50 hektar dengan 100 titik sensor, kemampuan sistem dalam menangani beban belum teruji. Kedua, parameter kesuburan tanah yang dimonitor baru terbatas pada kelembaban, pH, dan suhu. Unsur hara makro seperti nitrogen, fosfor, dan kalium tidak diukur karena keterbatasan sensor yang terjangkau, padahal informasi tersebut sangat dibutuhkan petani untuk pemupukan presisi. Ketiga, sistem belum terintegrasi dengan aktuator seperti pompa irigasi otomatis. Petani masih harus bertindak secara manual meskipun sudah mendapat peringatan. Keempat, penelitian ini menggunakan data simulasi dari perangkat virtual, bukan data dari sensor nyata yang terpasang di kebun buah naga. Kondisi lapangan yang sebenarnya (gangguan sinyal, debu, kelembaban tinggi, suhu ekstrem) belum diujikan. Kelima, aspek keamanan siber seperti enkripsi data *end-to-end* dan perlindungan terhadap serangan *Distributed Denial of Service* (DDoS) belum menjadi fokus. Untuk implementasi publik yang terhubung ke internet, hal ini perlu mendapat perhatian serius.

5. KESIMPULAN DAN SARAN

Penelitian ini telah berhasil merancang dan membangun arsitektur *Backend* IoT berbasis web untuk sistem monitoring kesuburan tanah pada budidaya buah naga, dengan ruang lingkup terbatas pada aspek perangkat lunak. Arsitektur yang dikembangkan menggunakan *framework* Laravel 10 dengan pola MVC menghasilkan delapan *endpoint* RESTful API yang seluruhnya berfungsi sesuai spesifikasi, serta skema basis data MySQL yang terdiri atas empat tabel utama *devices*, *sensor_readings*, *thresholds*, *notifications* dalam bentuk normal ketiga. Dashboard berbasis PWA yang dibangun dapat diinstal pada perangkat mobile dan menyediakan

tampilan *real-time*, grafik historis, serta notifikasi *push* dengan latensi rata-rata 89 milidetik. Pengujian *blackbox* terhadap delapan skenario fungsional menunjukkan tingkat keberhasilan 100%, yang menegaskan bahwa sistem telah memenuhi seluruh kebutuhan fungsional. Evaluasi kinerja menghasilkan waktu tanggap rata-rata 340 milidetik untuk pengiriman data sensor melalui *endpoint* POST, masih jauh di bawah batas toleransi 1 detik untuk sistem *real-time*. Uji keandalan membuktikan bahwa sistem mampu beroperasi dengan tingkat keberhasilan pengiriman data 99,87% pada uji beban 20 perangkat virtual dan *uptime* 99,95% selama 30 hari, tanpa kehilangan data berkat mekanisme *retry* pada sisi perangkat. Dengan demikian, arsitektur yang dihasilkan layak untuk diimplementasikan pada kebun buah naga skala menengah.

Untuk pengembangan selanjutnya, disarankan untuk mengganti protokol komunikasi dari HTTP menjadi MQTT pada lapisan pengiriman data sensor guna mengurangi latensi dan konsumsi daya perangkat. Penambahan modul analisis prediktif berbasis *machine learning* perlu dipertimbangkan agar sistem tidak hanya memonitor tetapi juga memprediksi tren kesuburan tanah berdasarkan data historis. Integrasi dengan aktuator seperti pompa irigasi otomatis dan dispenser pupuk akan menyempurnakan sistem menjadi kendali tertutup (*closed-loop control*). Uji coba lapangan langsung pada kebun buah naga dengan sensor nyata selama minimal satu musim tanam 4 bulan sangat diperlukan untuk memvalidasi arsitektur dalam kondisi lingkungan sebenarnya. Terakhir, aspek keamanan siber seperti enkripsi *end-to-end* dan perlindungan terhadap serangan DDoS perlu menjadi prioritas pada penelitian berikutnya sebelum sistem diterapkan secara publik.

DAFTAR PUSTAKA

- [1] G. Devira Ramady and A. Ghea Mahardika, "Analisis Uji Implementasi Smart Agriculture System Pada Lahan Terbatas Rumah di Wilayah Perkotaan Berbasis Kontrol Menggunakan Mikrokontroler Arduino Uno," *Smart Comp: Jurnalnya Orang Pintar Komputer*, vol. 10, no. 2, pp. 54–60, Jun. 2021, doi: 10.30591/smartcomp.v10i2.2310.
- [2] G. Saha, F. Shahrin, F. H. Khan, M. M. Meshkat, and A. A. M. Azad, "Smart IoT-driven precision agriculture: Land mapping, crop prediction, and irrigation system," *PLoS One*, vol. 20, no. 3, p. e0319268, Mar. 2025, doi: 10.1371/journal.pone.0319268.
- [3] A. W. Hakis, A. L. Arda, and A. Jalil, "IoT-based Soil Nutrient Monitoring and Control Using Fuzzy Logic and Multi-Modal Sensor Integration," *Journal of Applied Informatics and Computing*, vol. 9, no. 5, pp. 2736–2745, Oct. 2025, doi: 10.30871/jaic.v9i5.10575.
- [4] A.-T. Tran, T. H. Nguyen, H. Truong, N. Q. Pham, and T. Ho, "Development of an IoT-Based Real-Time Monitoring System for Light Intensity, Temperature, and Humidity in Dragon Fruit Farms," *Journal of Electronic & Information Systems*, vol. 8, no. 1, pp. 15–29, Jan. 2026, doi: 10.30564/jeis.v8i1.12853.
- [5] H. Benyezza, M. Bouhedda, R. Kara, and S. Rebouh, "Smart platform based on IoT and WSN for monitoring and control of a greenhouse in the context of precision agriculture," *Internet of Things*, vol. 23, p. 100830, Oct. 2023, doi: 10.1016/j.iot.2023.100830.
- [6] M. D. Agustiniingsih, P. C. A. Lestari, R. M. Savitrah, A. Fahmi, and H. -, "Nogo Osing Apps: Aplikasi Smart Farming Buah Naga Berbasis IoT," *Jurnal Teknologi Informasi dan Ilmu Komputer*, vol. 11, no. 1, pp. 145–154, Feb. 2024, doi: 10.25126/jtiik.20241117683.
- [7] A. Ehsan, M. A. M. E. Abuhaliqa, C. Catal, and D. Mishra, "RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions," *Applied Sciences*, vol. 12, no. 9, p. 4369, Apr. 2022, doi: 10.3390/app12094369.
- [8] E. M. Pechlivani, A. Papadimitriou, S. Pemas, G. Ntinis, and D. Tzovaras, "IoT-Based Agro-Toolbox for Soil Analysis and Environmental Monitoring," *Micromachines (Basel)*, vol. 14, no. 9, p. 1698, Aug. 2023, doi: 10.3390/mi14091698.
- [9] J. Xu, B. Gu, and G. Tian, "Review of agricultural IoT technology," *Artificial Intelligence in Agriculture*, vol. 6, pp. 10–22, 2022, doi: 10.1016/j.aiia.2022.01.001.
- [10] S. V. Gaikwad, A. D. Vibhute, K. V. Kale, and S. C. Mehrotra, "An innovative IoT based system for precision farming," *Comput. Electron. Agric.*, vol. 187, p. 106291, Aug. 2021, doi: 10.1016/j.compag.2021.106291.
- [11] V. Y. P. Ardhana, M. T. Hidayat, M. Jannah, S. Sumiati, P. Rini, and N. Sari, "Implementasi RESTful API Pada Laravel dan Simulator IoT Wokwi Untuk Pengukuran Suhu dan Kelembaban Menggunakan Metode Waterfall," *Arcitech: Journal of Computer Science and Artificial Intelligence*, vol. 3, no. 2, p. 93, Dec. 2023, doi: 10.29240/arcitech.v3i2.9334.
- [12] Y. Wu, Z. Yang, and Y. Liu, "Internet-of-Things-Based Multiple-Sensor Monitoring System for Soil Information Diagnosis Using a Smartphone," *Micromachines (Basel)*, vol. 14, no. 7, p. 1395, Jul. 2023, doi: 10.3390/mi14071395.
- [13] A. Mhatre and S. Mali, "Progressive Web Applications, a New Way for Faster Testing of Mobile Application Products," in *2023 3rd Asian Conference on Innovation in Technology (ASIANCON)*, IEEE, Aug. 2023, pp. 1–6. doi: 10.1109/ASIANCON58793.2023.10269806.
- [14] J. Bastos, P. M. Shepherd, P. Castillejo, M. S. Emeterio, V. H. Díaz, and J. Rodriguez,

- “Location-Based Data Auditing for Precision Farming IoT Networks,” in *2021 IEEE 26th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, IEEE, Oct. 2021, pp. 1–6. doi: 10.1109/CAMAD52502.2021.9617765.
- [15] P. K. Kashyap, S. Kumar, A. Jaiswal, M. Prasad, and A. H. Gandomi, “Towards Precision Agriculture: IoT-Enabled Intelligent Irrigation Systems Using Deep Learning Neural Network,” *IEEE Sens. J.*, vol. 21, no. 16, pp. 17479–17491, Aug. 2021, doi: 10.1109/JSEN.2021.3069266.
- [16] S. A. Bhat and N.-F. Huang, “Big Data and AI Revolution in Precision Agriculture: Survey and Challenges,” *IEEE Access*, vol. 9, pp. 110209–110222, 2021, doi: 10.1109/ACCESS.2021.3102227.
- [17] P. R. Pavan, M. G. Kerutagi, M. B. N. Naika, D. P. Prakasha, and V. K. Kumar, “Potentiality of dragon fruit for climate resilience and nutritional security: a critical review,” *Agric. Food Secur.*, vol. 14, no. 1, p. 56, Apr. 2026, doi: 10.1186/s40066-025-00544-1.
- [18] S. A. Astaneh, S. Shah Heydari, S. Taghavi Motlagh, and A. Izaddoost, “Trade-offs between Risk and Operational Cost in SDN Failure Recovery Plan,” *Future Internet*, vol. 14, no. 9, p. 263, Sep. 2022, doi: 10.3390/fi14090263.
- [19] M. Hardini, V. Agarwal, D. Apriani, I. A. Widjaya, E. Setiawaty, and N. Nurasiatiah, “Application of Database Normalization in Increasing Data Storage Efficiency,” *International Transactions on Artificial Intelligence (ITALIC)*, vol. 3, no. 2, pp. 201–211, May 2025, doi: 10.33050/italic.v3i2.799.
- [20] A. Biørn-Hansen, T. A. Majchrzak, and T.-M. Grønli, “Progressive Web Apps: The Possible Web-native Unifier for Mobile Development,” in *Proceedings of the 13th International Conference on Web Information Systems and Technologies*, SCITEPRESS - Science and Technology Publications, 2017, pp. 344–351. doi: 10.5220/0006353703440351.
- [21] I. Ismail and J. Efendi, “Black-Box Testing : Analisis Kualitas Aplikasi Source Code Bank Programming,” *Jurnal JTIK (Jurnal Teknologi Informasi dan Komunikasi)*, vol. 4, no. 2, p. 1, Dec. 2020, doi: 10.35870/jtik.v5i1.148.
- [22] K. G. Ayu, D. W. Sari, A. Zein, and R. S. Fadila, “Pengujian Black-Box Dan Analisa Kualitas ISO 25010 Pada Aplikasi HESTI,” *CESS (Journal of Computer Engineering, System and Science)*, vol. 9, no. 2, p. 710, Jul. 2024, doi: 10.24114/cess.v9i2.61621.
- [23] K. K. Raghuvanshi, A. Agarwal, K. Jain, and V. B. Singh, “A time-variant fault detection software reliability model,” *SN Appl. Sci.*, vol. 3, no. 1, p. 18, Jan. 2021, doi: 10.1007/s42452-020-04015-z