PENGEMBANGAN BACKEND UNTUK APLIKASI MONITORING REALTIME TONGKAT PINTAR IOT BERBASIS MQTT DAN NODE.JS

(Skripsi)

Oleh

ALVIN REIHANSYAH MAKARIM

2115061083

FAKULTAS TEKNIK
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG
2025

PENGEMBANGAN BACKEND UNTUK APLIKASI MONITORING REALTIME TONGKAT PINTAR IOT BERBASIS MQTT DAN NODE.JS

Oleh

ALVIN REIHANSYAH MAKARIM

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

ABSTRAK

PENGEMBANGAN BACKEND UNTUK APLIKASI MONITORING REALTIME TONGKAT PINTAR IOT BERBASIS MQTT DAN NODE.JS

Oleh

ALVIN REIHANSYAH MAKARIM

Penyandang tunanetra di Indonesia masih menghadapi keterbatasan dalam mobilitas, terutama ketika harus bepergian secara mandiri. Tongkat konvensional belum mampu menjawab kebutuhan navigasi modern yang membutuhkan solusi lebih aman, *real-time*, dan terintegrasi. Penelitian ini bertujuan untuk mengembangkan sistem *backend* pada aplikasi monitoring tongkat pintar berbasis *Internet of Things* (IoT), yang mendukung deteksi rintangan, pelacakan lokasi, serta pengiriman notifikasi darurat secara *real-time*. *Backend* dikembangkan menggunakan Node.js dengan *framework* Express.js, broker MQTT Mosquitto sebagai penghubung komunikasi IoT, basis data MySQL untuk penyimpanan informasi, Redis untuk manajemen sesi dan autentikasi, serta Firebase Cloud Messaging (FCM) untuk notifikasi. Metode yang digunakan adalah *Rapid Application Development* (RAD) dengan tahapan *requirements planning*, *user design*, *construction*, dan *cutover*. Sistem diuji menggunakan metode *black-box* dan *load testing* untuk menilai fungsionalitas serta performa. Hasil penelitian menunjukkan bahwa *backend* mampu mengelola komunikasi data IoT secara stabil, memberikan notifikasi darurat tepat waktu, serta mendukung pelacakan lokasi pengguna secara *real-time* dengan tingkat respons yang baik. Dengan demikian, sistem yang dikembangkan dapat menjadi solusi inovatif untuk meningkatkan keamanan dan kemandirian penyandang tunanetra.

Kata kunci: *Internet of Things*, Tongkat Pintar, *Backend*, MQTT, Node.js, *Realtime Monitoring*

ABSTRACT

BACKEND DEVELOPMENT FOR IOT SMART STICK REALTIME MONITORING APPLICATION BASED ON MQTT AND NODE.JS

By

ALVIN REIHANSYAH MAKARIM

Visually impaired individuals in Indonesia continue to face mobility challenges, particularly when traveling independently. Conventional canes have not fully addressed modern navigation needs that require safer, real-time, and integrated solutions. This research aims to develop a backend system for a smart stick monitoring application based on the Internet of Things (IoT), supporting obstacle detection, location tracking, and emergency notifications in real time. The backend was developed using Node.js with the Express.js framework, the Mosquitto MQTT broker for IoT communication, MySQL as the main database, Redis for session and authentication management, and Firebase Cloud Messaging (FCM) for notifications. The Rapid Application Development (RAD) method was applied, consisting of the stages of requirements planning, user design, construction, and cutover. The system was tested using black-box and load testing methods to evaluate functionality and performance. The results show that the backend system successfully manages IoT data communication stably, delivers emergency notifications promptly, and supports real-time user location tracking with excellent response performance. Therefore, the developed system serves as an innovative solution to enhance the safety and independence of visually impaired individuals.

Keywords: Internet of Things, Smart Stick, Backend, MQTT, Node.js, Realtime Monitoring

Judul Skripsi	:	PENGEMBANGAN BACKEND UNTUK APLIKASI MONITORING REALTIME TONGKAT PINTAR IOT BERBASIS MQTT DAN NODE.JS
Nama Mahasiswa	:	Alvin Reihansyah Makarim
Nomor Pokok Mahasiswa	:	2115061083
Program Studi	:	Teknik Informatika
Jurusan	:	Teknik Elektro
Fakultas	:	Teknik

MENYETUJUI

1. Komisi Pembimbing

Pembimbing Utama	Pembimbing Pendamping
Ir. Gigih Forda Nama, S.T., M.T.I., IPM	Deny Budiyanto, S. Kom., M.T.
NIP. 198307122008121003	NIP. 199112082019031011

2. Mengetahui

Ketua Jurusan Teknik Elektro	Ketua Program Studi Teknik Informatika
Herlinawati, S.T., M.T.	Yessi Mulyani, S.T., M.T.
NIP. 197103141999032001	NIP. 197312262000122001

MENGESAHKAN

1. Tim Penguji

Ketua : **Ir. Gigih Forda Nama, S.T., M.T.I., IPM** .....

Sekretaris : **Deny Budiyanto, S. Kom., M.T.** .....

Penguji : **Wahyu Eko Sulistiono, S.T., M.Sc.** .....

2. Dekan Fakultas Teknik

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

Tanggal Lulus Ujian Skripsi : 28 Oktober 2025

SURAT PERNYATAAN

Dengan ini saya menyatakan bahwa skripsi berjudul "Pengembangan Backend untuk Aplikasi Monitoring Realtime Tongkat Pintar IoT Berbasis MQTT dan Node.js" sepenuhnya merupakan hasil karya saya sendiri. Apabila di kemudian hari terbukti pernyataan ini tidak benar, saya siap menerima sanksi sesuai dengan ketentuan hukum yang berlaku.

Bandar Lampung, 28 Oktober 2025

Penulis,

Alvin Reihansyah Makarim

NPM. 2115061083

RIWAYAT HIDUP

Penulis dilahirkan di Bandar Lampung pada tanggal 8 Agustus 2002, sebagai anak pertama dari pasangan Bapak Erwin dan Ibu Nurrahmi. Penulis menyelesaikan pendidikan formal di SD Negeri 2 Metro pada tahun 2015, kemudian melanjutkan ke SMP Negeri 4 Metro dan lulus pada tahun 2018, serta menamatkan pendidikan menengah atas di SMA Negeri 1 Metro pada tahun 2021. Pada tahun 2021, penulis diterima sebagai mahasiswa Program Studi Teknik Informatika Fakultas Teknik Universitas Lampung melalui jalur seleksi SBMPTN. Selama masa perkuliahan, penulis aktif berpartisipasi dalam berbagai kegiatan, antara lain:

1. Mengikuti kegiatan Studi Independen Bersertifikat dari Kementerian Pendidikan dan Budaya di mitra Bangkit Akademi pada tahun 2023.
2. Mengikuti Penelitian MBKM terkait pengumpulan *dataset* tanaman dan Pemberdayaan *Machine Learning* terhadap klasifikasi jenis tumbuhan pada tahun 2024.

MOTTO

"Terbentur, terbentur, terbentuk"

(Tan Malaka)

"*Our greatest glory is not in never falling, but in rising every time we fall.*"

(Confucius)

"Bukan tentang seberapa cepat sampai, tapi tentang seberapa kuat bertahan dan terus melangkah."

(Penulis)

PERSEMBAHAN

Dengan penuh rasa syukur kepada Allah SWT atas limpahan rahmat, karunia, dan petunjuk-Nya, karya sederhana ini saya persembahkan kepada:

Kedua orang tua tercinta, yang telah menjadi sumber kekuatan, kasih sayang, dan doa dalam setiap langkah hidup saya.

Keluarga besar yang selalu memberi semangat dan dukungan tanpa henti.

Sahabat-sahabat seperjuangan yang senantiasa hadir dalam suka dan duka.

Serta almamater kebanggaan, Universitas Lampung, tempat saya menimba ilmu dan pengalaman berharga.

SANWACANA

Puji syukur penulis panjatkan ke hadirat Allah SWT atas limpahan rahmat, taufik, dan hidayah-Nya sehingga penulis dapat menyelesaikan penyusunan skripsi yang berjudul **"**Pengembangan Backend untuk Aplikasi Monitoring Realtime Tongkat Pintar IoT Berbasis MQTT dan Node.js**"** dengan baik.

Penyusunan skripsi ini tidak terlepas dari dukungan, bimbingan, serta bantuan berbagai pihak yang telah memberikan kontribusi, baik dalam bentuk moril maupun materil. Oleh karena itu, dengan penuh rasa hormat dan ketulusan, penulis menyampaikan ucapan terima kasih dan penghargaan yang sebesar-besarnya kepada:

1. Kedua orang tua tercinta beserta keluarga besar yang senantiasa memberikan doa, kasih sayang, dukungan, serta motivasi yang tiada henti kepada penulis, baik secara moril maupun spiritual, sehingga penulis mampu menyelesaikan studi dan penelitian ini dengan penuh semangat.;
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. Ibu Yessi Mulyani, S.T., M.T., selaku Ketua Program Studi Teknik Informatika Fakultas Teknik Universitas Lampung, yang telah memberikan kesempatan, bimbingan, serta dorongan kepada penulis dalam menyelesaikan skripsi ini.;
5. Ibu Ir. Titin Yulianti, S.T., M.Eng., selaku Dosen Pembimbing Akademik, yang telah memberikan bimbingan, perhatian, serta motivasi selama penulis menjalani masa perkuliahan.;
6. Bapak Ir. Gigih Forda Nama, S.T., M.T.I., IPM, selaku Dosen Pembimbing Utama, yang dengan sabar, telaten, dan penuh perhatian telah memberikan

arahan, masukan, serta ilmu yang sangat berharga selama proses penelitian dan penulisan skripsi ini.;

7. Bapak Deny Budiyanto, S.Kom., M.T., selaku Dosen Pembimbing Pendamping, yang telah meluangkan waktu, tenaga, dan pikiran untuk memberikan bimbingan, saran, serta dukungan dalam penyempurnaan penelitian ini.;
8. Bapak Wahyu Eko Sulistiono, S.T., M.Sc., selaku Dosen Penguji, yang telah memberikan kritik, saran, dan masukan yang membangun demi penyempurnaan hasil penelitian ini.;
9. Seluruh Dosen Program Studi Teknik Informatika Fakultas Teknik Universitas Lampung, yang telah memberikan ilmu, wawasan, dan pengalaman berharga selama masa perkuliahan.;
10. Rekan satu tim penelitian, Muhkito Afif dan Pandu Wijaya, atas kerja sama yang solid, semangat kebersamaan, serta kontribusinya dalam pengembangan sistem tongkat pintar berbasis IoT ini.;
11. Teman-teman dalam grup "Teh Kotak", serta seluruh sahabat Teknik Informatika 2021 yang tidak dapat disebutkan satu per satu, atas kebersamaan, dukungan, dan semangat yang diberikan selama masa perkuliahan hingga penyusunan skripsi ini.;

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna, sehingga kritik dan saran yang membangun sangat diharapkan. Semoga karya ini dapat memberikan manfaat serta menjadi referensi bagi pengembangan penelitian di bidang *Internet of Things* dan komputasi *backend*. Penulis juga berharap seluruh pihak yang telah memberikan dukungan dan bantuan selama proses penyusunan skripsi ini senantiasa mendapatkan limpahan rahmat dan balasan kebaikan dari Allah SWT.

Bandar Lampung, 28 Oktober 2025
Penulis,

Alvin Reihansyah Makarim
NPM. 2115061083

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

I. PENDAHULUAN

1.1. Latar Belakang

Penyandang tunanetra di Indonesia menghadapi berbagai tantangan dalam menjalani aktivitas sehari-hari, khususnya ketika harus bepergian secara mandiri ke lingkungan baru. Mobilitas menjadi salah satu kendala utama yang sering kali menimbulkan kekhawatiran, baik bagi individu tunanetra maupun keluarga mereka. Saat ini, alat bantu yang tersedia, seperti tongkat konvensional, memberikan manfaat signifikan, tetapi belum sepenuhnya menjawab kebutuhan mobilitas modern yang memerlukan solusi lebih inovatif.

Berdasarkan data Kementerian Kesehatan RI tahun 2023 seperti yang dikutip dalam [1], jumlah penyandang tunanetra di Indonesia mencapai 1,5% dari total populasi penduduk, atau sekitar 4 juta jiwa. Angka ini menunjukkan tingginya kebutuhan akan solusi alat bantu yang dapat mendukung mobilitas dan kemandirian penyandang tunanetra dalam menjalani aktivitas sehari-hari. Data ini juga menggarisbawahi pentingnya pengembangan teknologi yang dapat membantu penyandang tunanetra untuk lebih mandiri dan merasa aman saat beraktivitas.

Jumlah penyandang tunanetra yang ada di Kota Metro cukup signifikan. Berdasarkan hasil wawancara dengan salah satu pengurus Persatuan Tunanetra Indonesia (Pertuni), tercatat sekitar 40 anggota aktif di wilayah tersebut. Namun, infrastruktur kota ini masih kurang mendukung kebutuhan kaum difabel tunanetra, seperti minimnya *guiding block*, sehingga membatasi mobilitas mereka. Oleh karena itu, diperlukan solusi yang efektif untuk membantu mereka menjalani aktivitas sehari-hari dengan lebih aman dan mandiri.

Internet of Things (IoT) menawarkan solusi yang menjanjikan untuk mengatasi tantangan ini. IoT memungkinkan pengembangan alat bantu yang dapat mendeteksi rintangan berbasis sensor ultrasonik, melacak lokasi secara *real-time* menggunakan

Global Positioning System (GPS), dan menyediakan tombol darurat untuk mengirimkan sinyal darurat dalam keadaan kritis. Dengan fitur ini, alat bantu tidak hanya meningkatkan kemandirian penyandang tunanetra tetapi juga memberikan ketenangan bagi keluarga yang dapat memantau lokasi mereka melalui aplikasi seluler.

Penggunaan teknologi modern seperti IoT, GPS, dan sensor ultrasonik telah menunjukkan potensi besar dalam memecahkan masalah ini. Penelitian sebelumnya telah membuktikan efektivitas perangkat berbasis IoT dalam meningkatkan akurasi dan presisi deteksi rintangan, serta memungkinkan pemantauan lokasi secara *real-time*. Sebagai contoh, salah satu penelitian yang menggunakan teknologi GPS dan sensor ultrasonik pada alat bantu tunanetra telah mencapai akurasi sebesar 99,99% dan presisi 98,65%, menunjukkan peningkatan signifikan dibandingkan alat bantu konvensional [2].

Pengembangan alat bantu berbasis IoT ini bertujuan untuk menjawab kebutuhan spesifik penyandang tunanetra, seperti deteksi rintangan, pelacakan lokasi, dan respons cepat dalam situasi darurat. Dalam penelitian ini, fokus utama adalah pengembangan sistem *backend* untuk mendukung fungsi-fungsi tersebut, termasuk implementasi server Message Queuing Telemetry Transport (MQTT) untuk komunikasi data *real-time*, pengembangan Application Programming Interface (API), serta pengelolaan basis data untuk menyimpan informasi lokasi dan notifikasi darurat. Dengan pendekatan ini, alat bantu yang dirancang diharapkan menjadi solusi inovatif dan inklusif untuk mendukung kualitas hidup penyandang tunanetra di Indonesia.

1.2.Rumusan Masalah

Adapun rumusan masalah dalam penelitian ini adalah sebagai berikut :

1. Bagaimana merancang sistem *backend* untuk mendukung alat bantu tunanetra berbasis IoT yang dapat mendeteksi rintangan dan memberikan peringatan secara *real-time*?
2. Bagaimana mengimplementasikan teknologi komunikasi MQTT untuk memastikan pengiriman data lokasi dan notifikasi darurat yang stabil dan efisien?

3. Bagaimana memastikan integrasi yang efisien antara perangkat keras tongkat pintar dan aplikasi seluler untuk mendukung kebutuhan pengguna tunanetra dan pengawas?

1.3.Tujuan Penelitian

Adapun tujuan dalam penelitian ini adalah sebagai berikut :

1. Merancang sistem *backend* untuk mendukung alat bantu tunanetra berbasis IoT yang mampu memberikan peringatan secara *real-time*.
2. Mengimplementasikan teknologi komunikasi MQTT untuk memastikan pengiriman data lokasi dan notifikasi darurat secara stabil dan efisien.
3. Mengintegrasikan perangkat keras tongkat pintar dengan aplikasi seluler secara efisien untuk mendukung kebutuhan pengguna tunanetra dan pengawas.

1.4.Manfaat Penelitian

Adapun manfaat penelitian dalam penelitian ini adalah sebagai berikut :

1. Memudahkan mobilitas dan meningkatkan rasa aman melalui pelacakan lokasi *real-time*.
2. Memberikan kemudahan dalam memantau lokasi penyandang tunanetra secara *real-time* dan menerima notifikasi darurat jika terjadi situasi kritis.
3. Menjadi referensi untuk pengembangan sistem berbasis IoT dan teknologi GPS yang dapat diimplementasikan untuk membantu kelompok tunanetra.

1.5.Batasan Masalah

Adapun Batasan masalah dalam penelitian ini adalah sebagai berikut :

1. Penelitian ini berfokus pada pengembangan *backend* sistem, termasuk server MQTT, API menggunakan Node.js, dan basis data MySQL.
2. Penelitian ini tidak mencakup pengembangan *frontend* aplikasi seluler atau pengembangan perangkat keras IoT secara mendalam, yang merupakan tanggung jawab anggota tim lainnya.
3. Data yang diproses oleh sistem *backend* meliputi data lokasi (*latitude, longitude, timestamp*), data sensor (ketinggian air, jarak objek), dan notifikasi darurat yang dikirim oleh perangkat IoT.

1.6.Sistematika Penulisan

Adapun sistematika yang digunakan dalam penulisan penelitian ini adalah sebagai berikut:

BAB I: PENDAHULUAN

Bab ini berisi mengenai latar belakang penelitian yang menjelaskan dasar pengembangan alat bantu navigasi berbasis IoT untuk tunanetra, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, dan sistematika penulisan.

BAB II: TINJAUAN PUSTAKA

Bab ini memuat teori-teori dasar dan penelitian terdahulu yang menjadi referensi dalam pengembangan sistem.

BAB III: METODE PENELITIAN

Bab ini menjelaskan metode penelitian yang digunakan. Bab ini juga berisi mengenai waktu penelitian, tempat penelitian, alat penelitian, struktur tim, dan tahapan yang dilakukan dalam penelitian ini, meliputi analisis kebutuhan, perencanaan kebutuhan, desain sistem, dan pengujian sistem.

BAB IV: HASIL DAN PEMBAHASAN

Bab ini berisi hasil penelitian yang diperoleh berdasarkan implementasi dan pengujian sistem. Data yang diperoleh dari proses penelitian dipaparkan dan dibahas terkait sejauh mana sistem yang dikembangkan mampu menjawab rumusan masalah yang telah ditetapkan.

BAB V: KESIMPULAN

Bab ini memuat kesimpulan yang diambil berdasarkan hasil penelitian, yang disusun sesuai dengan tujuan penelitian dan rumusan masalah yang telah dirumuskan. Selain itu, diberikan saran yang ditujukan untuk pengembangan penelitian lebih lanjut agar sistem dapat lebih optimal dan bermanfaat.

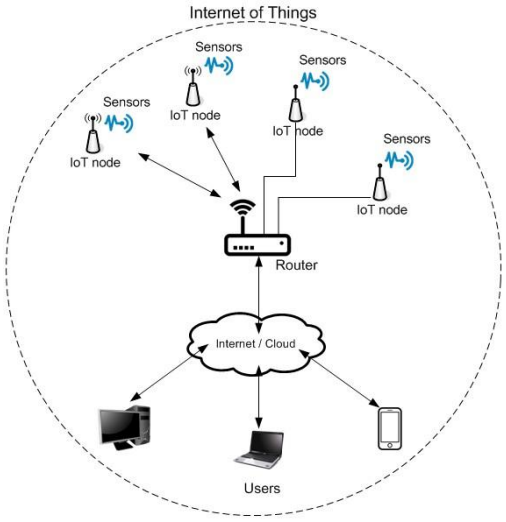
DAFTAR PUSTAKA

LAMPIRAN

II. TINJAUAN PUSTAKA

2.1.*Internet of Things* (IoT)

Internet of Things (IoT) merupakan paradigma teknologi yang memungkinkan objek-objek fisik untuk saling terhubung dan berkomunikasi melalui jaringan internet, sehingga dapat mengumpulkan dan bertukar data secara otomatis tanpa perlu kendali manusia [3]. Konsep IoT melibatkan integrasi sensor, aktuator, dan perangkat komunikasi yang bekerja bersama untuk mengumpulkan informasi dari lingkungan serta mengirimkannya ke platform pusat untuk analisis dan pengambilan keputusan. Penerapan IoT telah meluas di berbagai sektor, mulai dari kesehatan, industri, pertanian, hingga transportasi, di mana teknologi ini membantu meningkatkan efisiensi operasional dan kualitas hidup melalui automasi dan analitik data [4].



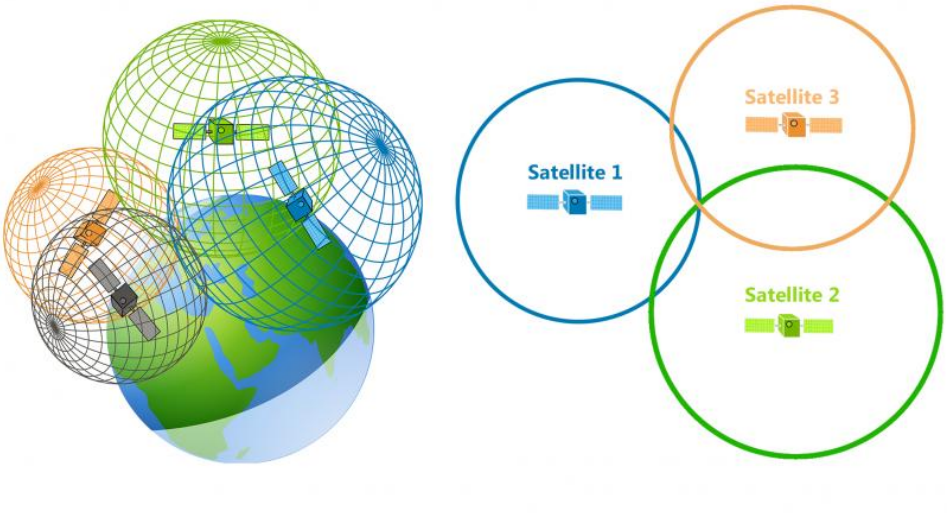
Gambar 2.1 Contoh Arsitektur *Internet of Things*

Dalam pengembangan alat bantu bagi tunanetra yang dilengkapi deteksi objek dan GPS, IoT memiliki peran krusial dalam memperoleh data koordinat garis bujur dan

latitude secara *real-time*. Informasi ini digunakan untuk menentukan lokasi tunanetra dan ditampilkan dalam aplikasi seluler, sehingga memungkinkan pemantauan jarak jauh oleh keluarga. Dengan demikian, keluarga dapat merasa lebih tenang ketika tunanetra bepergian sendiri. Selain itu, alat ini juga dilengkapi dengan tombol darurat yang berfungsi untuk mengirimkan notifikasi berisi titik lokasi penjemputan saat tunanetra mengalami situasi darurat.

2.2.*Global Positioning System* (GPS)

Global Positioning System (GPS) merupakan salah satu teknologi inti yang memanfaatkan sinyal satelit untuk menentukan posisi geografis dengan tingkat akurasi yang tinggi. Dalam sistem GPS, minimal empat satelit diperlukan untuk menghitung koordinat garis lintang, garis bujur, dan ketinggian [5].



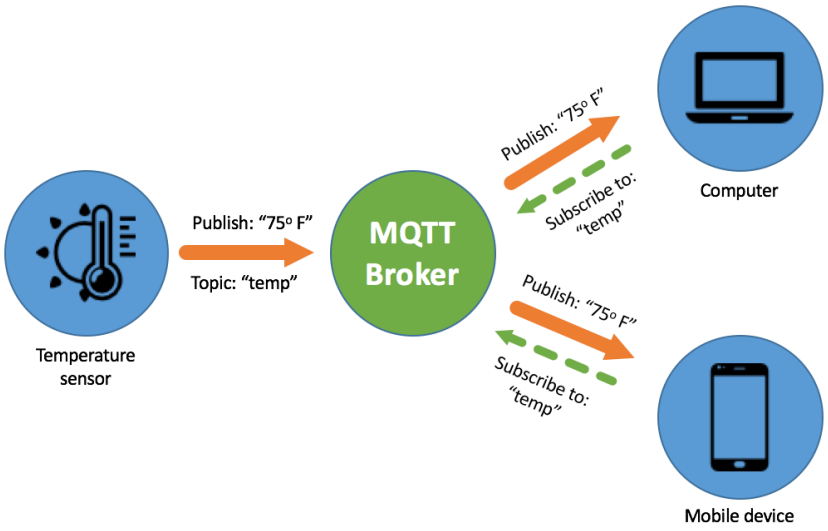
Gambar 2.2 Cakupan global sistem satelit untuk komunikasi dan navigasi

Teknologi ini telah diterapkan dalam berbagai bidang, terutama dalam pengembangan alat bantu untuk penyandang tunanetra, di mana akurasi dan kecepatan penyampaian data lokasi sangat krusial. Misalnya, penelitian yang dilakukan oleh Anggara dan Taufiq [6] mengembangkan sistem pelacakan berbasis GPS dalam rangka meningkatkan mobilitas penyandang tunanetra melalui integrasi dengan platform IoT. Selanjutnya, Pambudi dan Leonardo [7] merancang tongkat bantu inovatif yang mengombinasikan sensor giroskop, GPS, dan ultrasonik guna mendeteksi rintangan serta memantau pergerakan pengguna secara *real-time*,

sehingga memberikan peringatan dini dalam situasi kritis. Pada penelitian lainnya, Ahmed dkk. [8] mengintegrasikan GPS ke dalam tongkat pintar multifungsi untuk menyediakan data lokasi secara akurat melalui jaringan komunikasi IoT, yang memungkinkan pengawasan jarak jauh oleh keluarga atau pendamping. Dengan demikian, GPS tidak hanya berperan sebagai alat pelacak lokasi, tetapi juga sebagai komponen vital dalam mengembangkan teknologi bantu yang meningkatkan kemandirian dan keselamatan penyandang tunanetra. Integrasi GPS dengan protokol komunikasi seperti MQTT dan sensor pendukung lainnya memungkinkan sistem untuk mengirimkan data lokasi secara *real-time*, sehingga memberikan jaminan bahwa pengawasan dan respons terhadap keadaan darurat dapat dilakukan dengan cepat dan akurat.

2.3.*Message Queuing Telemetry Transport* (MQTT)

Message Queuing Telemetry Transport (MQTT) merupakan protokol komunikasi yang ringan dan efisien, didesain khusus untuk mendukung komunikasi *machine-to-machine* (M2M) serta IoT dalam kondisi jaringan dengan *bandwidth* terbatas dan latensi yang rendah. Protokol ini menggunakan model *publish/subscribe* yang mengandalkan broker sebagai perantara antara perangkat penerbit (*publisher*) dan perangkat penerima (*subscriber*), sehingga pesan yang diterbitkan ke sebuah topik secara otomatis didistribusikan kepada seluruh klien yang berlangganan topik tersebut.



Gambar 2.3 Ilustrasi penggunaan protokol MQTT

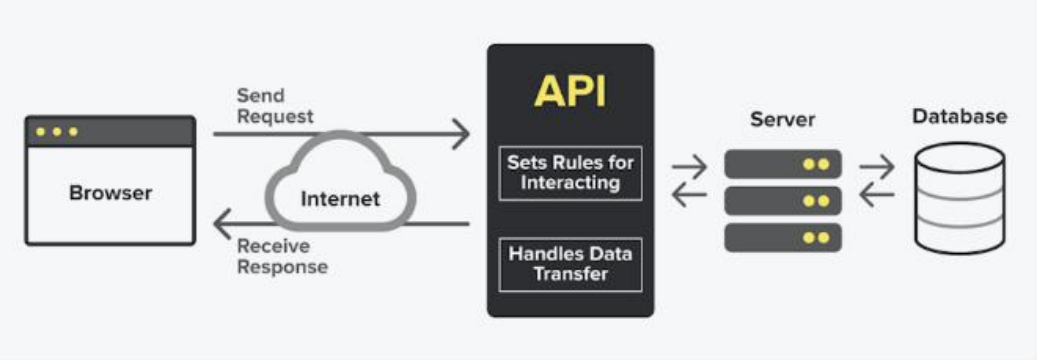
MQTT awalnya dikembangkan oleh IBM pada akhir 1990-an untuk mengatasi tantangan komunikasi jarak jauh pada sistem dengan sumber daya terbatas, dan sejak saat itu telah diadopsi secara luas di berbagai aplikasi IoT modern. Keunggulan utama MQTT terletak pada *overhead* yang minimal, kemudahan implementasi, dan fleksibilitas dalam mengatur *Quality of Service* (QoS) untuk menjamin pengiriman pesan secara andal sesuai dengan kebutuhan aplikasi—mulai dari pengiriman data sensor secara *real-time* hingga pemberian notifikasi darurat pada sistem monitoring. Sebagai contoh, penelitian yang dilakukan oleh Nugraha dkk. [9] membandingkan performa antara *Hypertext Transfer Protocol* (HTTP) dan MQTT pada aplikasi IoT dalam konteks sistem hidroponik, menunjukkan bahwa MQTT memiliki keunggulan dalam efisiensi sumber daya dan kecepatan transfer data. Selain itu, studi oleh Atmoko dkk. [10] menyebutkan bahwa protokol HTTP tidak dirancang untuk lingkungan jaringan yang berskala luas dan terus-menerus terhubung, sehingga dapat menyebabkan penurunan performa, khususnya dalam efisiensi penggunaan *bandwidth* dan daya tahan baterai. Dengan demikian, MQTT tidak hanya menjadi fondasi dalam infrastruktur komunikasi IoT modern, tetapi juga membuka peluang untuk pengembangan aplikasi-aplikasi canggih yang dapat meningkatkan efisiensi dan responsivitas sistem di berbagai bidang teknologi.

2.4.Eclipse Mosquitto

Eclipse Mosquitto merupakan broker MQTT *open source* yang dikembangkan oleh Eclipse Foundation dan dirancang untuk mendukung komunikasi *real-time* dalam sistem IoT. Broker ini memiliki arsitektur yang ringan serta hemat sumber daya, sehingga sangat cocok untuk perangkat dengan keterbatasan memori dan komputasi, dan mampu menangani koneksi dari banyak klien secara simultan. Mosquitto mendukung protokol MQTT versi 3.1 dan 3.1.1, sehingga memungkinkan penerapan berbagai tingkat QoS untuk menjamin keandalan pengiriman pesan, mulai dari aplikasi yang menuntut respons cepat hingga aplikasi dengan kebutuhan ketepatan data yang tinggi [11]. Berbagai studi telah mengevaluasi performa dan skalabilitas Eclipse Mosquitto, di mana broker ini menunjukkan latensi rendah dan *throughput* yang tinggi bila dibandingkan dengan broker MQTT lainnya, sehingga menjadikannya pilihan ideal untuk aplikasi monitoring *real-time* dan sistem automasi [12], [13], [14]. Fitur *bridging* yang

dimiliki Mosquitto juga mendukung distribusi pesan antar broker dalam jaringan yang lebih besar, memungkinkan pengembangan sistem terdistribusi yang efisien. Dengan kemampuan tersebut, Eclipse Mosquitto telah menjadi komponen penting dalam ekosistem IoT modern, menyediakan solusi komunikasi yang efisien, fleksibel, dan mudah diintegrasikan dengan berbagai layanan dan perangkat IoT, yang sangat relevan untuk pengembangan aplikasi berbasis IoT dalam konteks penelitian dan industri.

2.5.*Application Programming Interface* (API)



Gambar 2.4 Alur kerja API

Application Programming Interface (API) adalah kumpulan aturan, protokol, dan definisi yang memungkinkan komunikasi antara komponen perangkat lunak secara terstruktur tanpa harus mengungkap detail implementasi internalnya. API memainkan peran krusial dalam pengembangan sistem modern, terutama dalam arsitektur *microservices* dan aplikasi terdistribusi, di mana modularitas dan skalabilitas sangat penting. Dengan menggunakan API, pengembang dapat mengintegrasikan berbagai layanan dan sumber data dari sistem yang berbeda secara efisien.

Menurut Bloch [15], API yang baik harus mempertimbangkan kemudahan penggunaan, penamaan *endpoint* yang baik, dan dokumentasi yang jelas agar dapat digunakan secara efektif oleh developer. Bloch juga menambahkan bahwa perancangan API bukanlah aktivitas yang dilakukan secara individual. Desain API sebaiknya ditinjau oleh banyak pihak agar memperoleh masukan yang beragam, karena kemungkinan-kemungkinan yang tidak terpikirkan oleh perancang mungkin justru jelas bagi orang lain. Ia juga menekankan pentingnya kesederhanaan dalam

desain API, dengan prinsip *'when in doubt, leave it out'* yang mengarah pada pembuatan API yang minimalis namun tetap fungsional, karena fitur selalu dapat ditambahkan kemudian, tetapi sulit untuk dihapus tanpa mengganggu kompatibilitas.

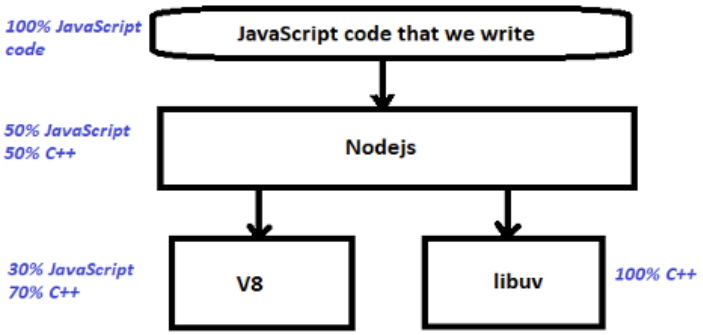
2.6.RESTful API

Representational State Transfer (REST) adalah gaya arsitektur perangkat lunak yang menetapkan sejumlah batasan dalam perancangan API. REST diciptakan sebagai panduan untuk mengatur komunikasi dalam sebuah jaringan yang kompleks seperti internet [16]. REST memanfaatkan metode standar HTTP seperti GET, POST, PUT, dan DELETE untuk melakukan operasi *Create, Read, Update, Delete* (CRUD) pada sumber daya yang diidentifikasi oleh Uniform Resource Locator (URL). Dalam konteks RESTful API, gaya arsitektur REST memberikan kerangka kerja yang intuitif di mana setiap permintaan HTTP memuat informasi yang diperlukan untuk pemrosesan tanpa perlu menyimpan status di sisi server. Gowda [17] menjelaskan bahwa kesederhanaan dan fleksibilitas REST telah menjadikannya banyak digunakan untuk pengembangan layanan web modern, yang memungkinkan aplikasi untuk berkomunikasi secara cepat dan efisien dalam lingkungan terdistribusi. Selain itu, beliau juga menjelaskan bahwa desain RESTful API yang baik tidak hanya mementingkan mengenai skalabilitas, tetapi juga perlu keamanan yang terjamin. Protokol *Hypertext Transfer Protocol Secure* (HTTPS) dan *Open Authorization* (OAuth) menjadi garda terdepan dalam hal keamanan. Oleh karena itu, dengan mengintegrasikan prinsip-prinsip REST, pengembang dapat menciptakan API yang mudah diimplementasikan, dikelola, dan dikonsumsi, sehingga mendukung inovasi dan pertumbuhan ekosistem digital secara menyeluruh.

2.7.Node.js

Node.js adalah *runtime environment* JavaScript yang memungkinkan pengeksekusian kode di sisi server. Node.js dibuat untuk mengatasi masalah kinerja pada platform yang berkaitan dengan waktu komunikasi jaringan, di mana terlalu banyak waktu dihabiskan untuk memproses permintaan dan respons web. Node.js memungkinkan JavaScript digunakan secara *end-to-end*, baik di sisi klien maupun

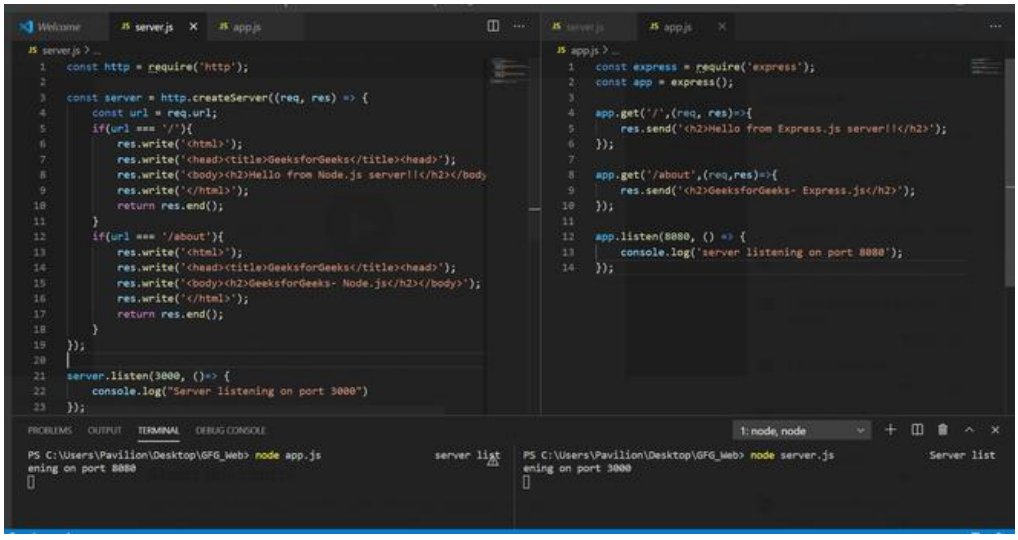
server. Node.js memanfaatkan mesin V8 dari Google sehingga mampu mengonversi JavaScript langsung ke dalam kode mesin.



Gambar 2.5 Struktur Internal Node.js

Selain itu, Node.js memiliki ekosistem yang luas melalui *Node Package Manager* (npm), yang menyediakan berbagai pustaka dan modul untuk mendukung pengembangan aplikasi secara modular dan efisien. Arsitektur *event-driven* yang diusung Node.js memungkinkan pengolahan permintaan secara cepat [18]. Dengan dukungan terhadap integrasi layanan komputasi awan serta kompatibilitas dengan berbagai *framework backend* seperti Express.js, Node.js telah menjadi salah satu pilihan utama dalam pengembangan aplikasi modern yang membutuhkan kinerja tinggi dan fleksibilitas dalam pengelolaan infrastruktur.

2.8.Express.js



Gambar 2.6 Perbandingan Penulisan Kode Antara Node.js (kiri) dan Express.js (kanan)

Express.js adalah sebuah *framework* kecil yang dibangun di atas fungsionalitas web server yang dimiliki oleh Node.js dengan tujuan untuk menyederhanakan API dan memudahkan pengembangan aplikasi *backend* dengan beberapa fitur yang dibawa olehnya. *Framework* ini menyediakan fitur-fitur penting seperti *routing*, *middleware*, dan manajemen *HTTP request/response*, sehingga memungkinkan pembuatan RESTful API dan aplikasi web secara cepat dan efisien. Ketika Node.js menyediakan *runtime*-nya, Express.js menyederhanakan pembuatan aplikasi *server-side*.

Pada Node.js, fitur *routing* tidak tersedia secara bawaan. Sebagai gantinya, developer perlu memeriksa URL dan metode dari setiap permintaan yang masuk secara manual untuk menentukan cara menanganinya. Oleh karena itu, logika *routing* harus diimplementasikan langsung dalam kode aplikasi, biasanya menggunakan pernyataan kondisional atau *framework* seperti Express.js yang menyediakan sistem *routing* yang lebih terstruktur. Penggunaan Express.js menyederhanakan implementasi *routing* dengan memberikan kemudahan dalam menentukan rute dan fungsi yang sesuai. Melalui fungsi bawaan Express, kita dapat langsung menetapkan nama rute serta jenis permintaan, seperti GET atau POST. Pendekatan ini mempercepat proses pengembangan, meningkatkan produktivitas, dan mengurangi kompleksitas. Berdasarkan hasil penelitian yang dilakukan oleh Amarulloh dkk [19], Express.js memiliki tingkat keberhasilan *request* yang tinggi dibandingkan *framework* Django dan Laravel, membuatnya penting dalam aplikasi yang membutuhkan *read-write* data yang cepat dan berulang.

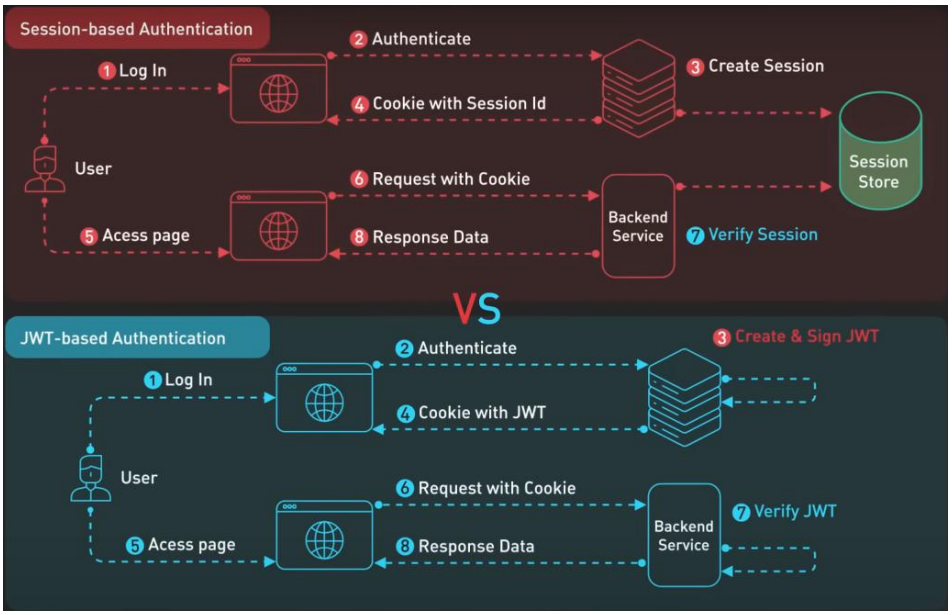
2.9.Firebase Cloud Messaging (FCM)

Firebase Cloud Messaging (FCM) adalah layanan pengiriman notifikasi dan pesan *real-time* yang dikembangkan oleh Google sebagai bagian dari platform Firebase. FCM memungkinkan pengembang untuk mengirimkan notifikasi *push* dan pesan data ke perangkat Android, iOS, dan web secara efisien, berkat infrastruktur *cloud* yang mendukung skalabilitas tinggi. Layanan ini menggunakan token registrasi untuk memastikan keamanan dan keakuratan pengiriman pesan, sehingga setiap perangkat hanya menerima pesan yang relevan. Selain itu, FCM mendukung komunikasi dua arah antara server dan klien, memungkinkan aplikasi untuk

memperoleh pembaruan secara instan dan responsif sesuai dengan kebutuhan aplikasi.

Keunggulan FCM terletak pada kemudahan integrasi dengan berbagai layanan Firebase dan Google Cloud, serta kemampuan pengelolaan pesan secara terpusat melalui konsol web yang intuitif. FCM menyediakan analisis kinerja pengiriman yang membantu pengembang memantau efektivitas notifikasi dan mengoptimalkan strategi komunikasi dengan pengguna. Menggunakan FCM, pengembang dapat mengirimkan notifikasi berdasarkan topik, yang didasarkan pada model *publish/subscribe*. Pengembang dapat menyusun pesan topik sesuai kebutuhan, dan FCM menangani perutean dan pengiriman pesan ke perangkat yang tepat [20].

2.10. *JSON Web Token* (JWT)



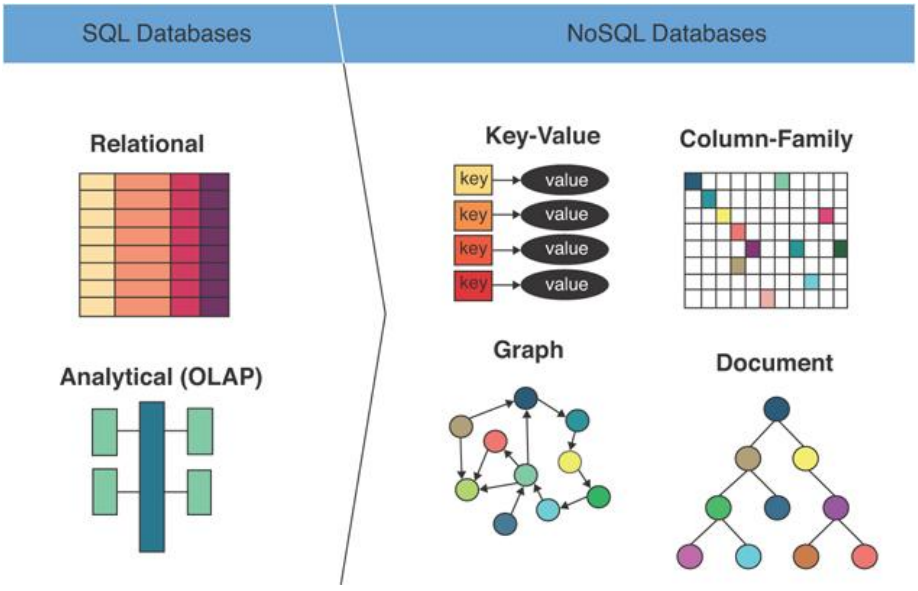
Gambar 2.7 Proses validasi JWT untuk akses sumber daya di server.

JSON Web Token (JWT) adalah standar terbuka yang dirancang untuk pertukaran informasi secara aman antara berbagai entitas melalui format *JavaScript Object Notation* (JSON) yang ringkas dan *self-contained*. JWT terdiri dari tiga bagian utama—*header*, *payload*, dan *signature*—yang masing-masing dienkode menggunakan Base64Url. *Header* mendefinisikan algoritma yang digunakan untuk penandatanganan, *payload* memuat klaim-klaim yang berisi informasi identitas atau metadata lainnya, dan *signature* dihasilkan dengan menandatangani kombinasi

header dan *payload* menggunakan kunci rahasia atau kunci publik-pribadi. Dengan sifatnya yang *stateless*, JWT memungkinkan sistem autentikasi tanpa perlu menyimpan informasi sesi di sisi server, sehingga mengurangi *overhead* penyimpanan dan meningkatkan skalabilitas.

Menurut Suryawanshi [21], implementasi JWT dalam konteks API memberikan keuntungan berupa kecepatan autentikasi serta efisiensi dalam distribusi token pada arsitektur *microservices* dan *stateless*. Selain itu, interoperabilitas yang ditawarkan oleh format JSON memudahkan integrasi antar sistem yang berbeda. Meskipun menawarkan banyak keuntungan, penerapan JWT harus disertai dengan perhatian khusus. Misalnya, mencabut token bisa menjadi sulit setelah token diterbitkan karena server secara *bawaan* tidak menyimpan sesi pengguna. Desain yang matang, seperti menggunakan token dengan masa berlaku pendek dan *refresh token*, dapat membantu mengatasi masalah ini.

2.11. Basis Data



Gambar 2.8 Perbandingan basis data SQL dan NoSQL

Basis data adalah kumpulan data yang terorganisir dan dikelola untuk memudahkan penyimpanan, pengambilan, dan pengelolaan informasi. Dalam konteks sistem informasi, basis data berfungsi sebagai fondasi untuk menyimpan data yang diperlukan oleh aplikasi, memungkinkan pengguna untuk melakukan operasi seperti pencarian, pembaruan, dan penghapusan data dengan efisien.

Ada berbagai jenis basis data, termasuk basis data relasional, non-relasional, dan basis data terdistribusi. Basis data relasional, yang paling umum digunakan, menyimpan data dalam tabel yang saling berhubungan, memungkinkan pengguna untuk melakukan *query* menggunakan *Structured Query Language* (SQL). Di sisi lain, basis data non-relasional, seperti *NoSQL*, dirancang untuk menangani data yang tidak terstruktur dan skala besar, memberikan fleksibilitas dalam pengelolaan data. Seperti yang dijelaskan oleh Sahatqija dkk. [22], pemilihan jenis basis data yang tepat sangat bergantung pada kebutuhan aplikasi dan karakteristik data yang akan dikelola.

2.12. MySQL

MySQL merupakan salah satu sistem manajemen basis data relasional (RDBMS) open-source yang paling populer dan banyak digunakan dalam berbagai aplikasi, mulai dari website hingga sistem *e-commerce* dan IoT. Kinerja dan skalabilitas MySQL telah menjadi fokus penelitian guna mengoptimalkan performa basis data dalam kondisi beban yang beragam. Menurut Ivan [23], optimasi performa MySQL dapat dicapai melalui beberapa teknik utama, seperti optimasi *query*, penggunaan indeks yang efisien, dan perancangan struktur tabel yang tepat. Teknik-teknik ini terbukti dapat mengurangi waktu respons sistem serta meningkatkan *throughput* basis data, sehingga MySQL mampu menangani volume transaksi yang tinggi secara lebih optimal. Secara keseluruhan, berbagai penelitian mengindikasikan bahwa MySQL memiliki fleksibilitas dan keandalan yang tinggi, asalkan didukung dengan strategi optimasi yang tepat sesuai dengan karakteristik dan kebutuhan aplikasi yang diimplementasikan.

2.13. Redis

Redis (*Remote Dictionary Server*) merupakan basis data *open source* yang menyimpan data langsung di dalam memori. Pendekatan ini memungkinkan proses akses dan pengambilan data berlangsung sangat cepat dibandingkan dengan basis data relasional [24]. Dalam pengembangan *backend*, Redis banyak dimanfaatkan sebagai media penyimpanan sementara, misalnya untuk menyimpan kode OTP, token autentikasi, atau data yang sering diakses. Dengan demikian, Redis dapat membantu mengurangi beban kerja pada basis data utama.

2.14. Rumus Haversine

Rumus Haversine digunakan untuk menghitung jarak lurus terpendek antara dua titik berdasarkan koordinat lintang dan bujur dari kedua titik tersebut [25]. Penelitian yang dilakukan oleh Saputra dkk. menunjukkan bahwa rumus ini cukup akurat untuk menghitungkan jarak [26]. Rumus Haversine dapat dijabarkan sebagai berikut:

$$\Delta lat = lat_2 - lat_1$$

$$\Delta lon = lon_2 - lon_1$$

$$a = sin^2\left(\frac{\Delta lat}{2}\right) + \cos\left(lat_1\right) \cdot cos(lat_2) \cdot sin^2\left(\frac{\Delta lon}{2}\right)$$

$$c = 2 \cdot atan2\left(\sqrt{a}, \sqrt{1-a}\right)$$

$$d = R \cdot c$$

Keterangan:

lat_1, lon_1 = koordinat titik pertama (dalam radian)

lat_2, lon_2 = koordinat titik kedua (dalam radian)

$d =$ jarak antara dua titik (km)

$R =$ radius bumi (6317 km)

Δlat = selisih lintang

Δlon = selisih bujur

R = jari-jari bumi (6371 km atau 6.371.000 m)

d = jarak antara kedua titik (km atau m)

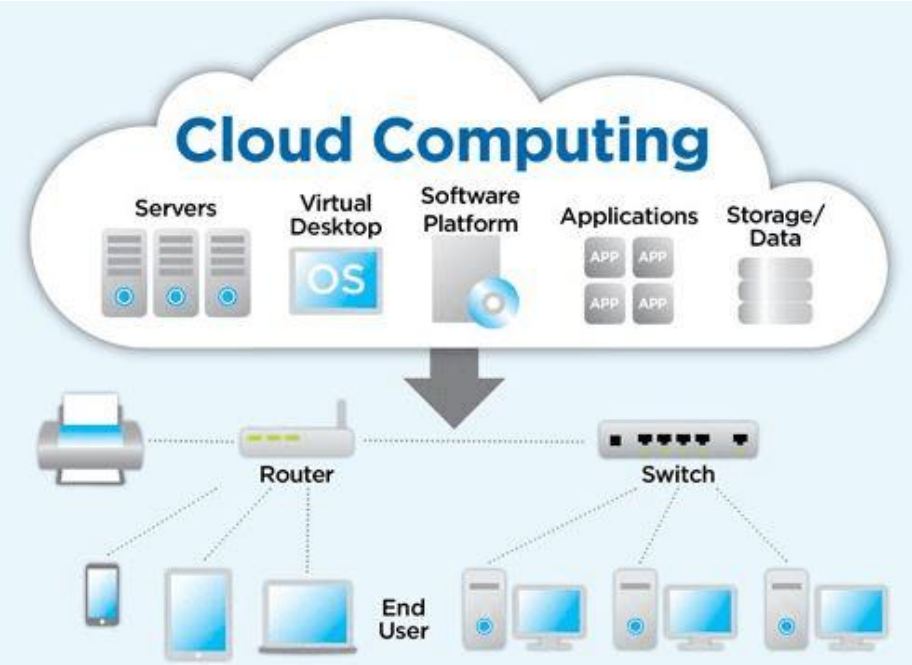
2.15. WebSocket

WebSocket merupakan protokol komunikasi berbasis TCP yang mendukung pertukaran data dua arah secara *real-time* melalui satu koneksi yang tetap terbuka. Dengan mekanisme ini, server dapat langsung mengirimkan data ke *client* tanpa menunggu permintaan, berbeda dengan pola komunikasi HTTP tradisional. Dalam

pengembangan aplikasi *backend*, WebSocket banyak digunakan untuk menangani kebutuhan komunikasi *real-time*, seperti layanan notifikasi, sistem *chat*, dan pemantauan data, sehingga memungkinkan sistem memberikan respons yang lebih cepat dan interaktif. Beberapa penelitian telah membandingkan perbedaan kecepatan antara HTTP dan WebSocket, salah satunya yang dilakukan oleh Maulana dkk. Pada penelitian ini, dibuktikan bahwa WebSocket mampu menghemat *transmit* dan *recieve data network* serta meningkatkan kecepatan komunikasi dua arah [27].

2.16. *Cloud Computing*

Cloud computing merupakan model penyediaan sumber daya komputasi—seperti penyimpanan, pemrosesan, dan aplikasi—secara elastis dan *on-demand* melalui internet.



Gambar 2.9 Arsitektur *Cloud Computing*

Teknologi ini mengintegrasikan konsep virtualisasi, komputasi terdistribusi, dan manajemen sumber daya untuk menghadirkan layanan yang skalabel serta efisien secara biaya. *Cloud computing* telah merevolusi cara organisasi mengelola infrastruktur TI mereka dengan mengalihkan beban investasi dari perangkat keras dan pemeliharaan ke model layanan yang lebih fleksibel. Dalam studi oleh Zhang

dkk. [28], *cloud computing* dijelaskan sebagai paradigma yang memungkinkan penyediaan layanan TI secara dinamis dengan model bayar sesuai pemakaian, yang memberikan keuntungan signifikan dalam hal fleksibilitas dan efisiensi biaya. Selain itu, penelitian ini juga mengidentifikasi tantangan utama dalam penerapan *cloud computing*, seperti pengelolaan sumber daya yang optimal, pengendalian beban kerja yang dinamis, serta isu keamanan dan privasi data. Tantangan-tantangan ini memicu inovasi dalam teknik virtual isasi dan algoritma manajemen sumber daya yang bertujuan untuk meningkatkan kinerja dan keandalan sistem *cloud*. Secara keseluruhan, literatur mengindikasikan bahwa meskipun *cloud computing* menawarkan manfaat signifikan bagi efisiensi operasional dan fleksibilitas, keberhasilan implementasinya sangat bergantung pada strategi optimasi dan mitigasi risiko terkait keamanan serta pengelolaan sumber daya.

2.17. GitHub

GitHub adalah platform kolaborasi pengembangan perangkat lunak yang menyediakan layanan pengendalian versi berbasis Git serta berbagai fitur pendukung seperti pelacakan isu, manajemen proyek, dan integrasi dengan alat-alat *Continuous Integration/Continuous Deployment* (CI/CD). Penggunaan GitHub telah menjadi sumber data empiris yang kaya dalam penelitian rekayasa perangkat lunak, terutama untuk mengkaji tantangan dan praktik terbaik dalam pengembangan perangkat lunak sumber terbuka. Dengan diperkenalkannya GitHub Actions pada tahun 2019, platform ini semakin mengoptimalkan proses automasi dalam CI/CD [29].

2.18. *Blackbox Testing*

Black-box testing merupakan metode pengujian perangkat lunak yang berfokus pada evaluasi fungsi sistem berdasarkan spesifikasi yang telah ditetapkan, tanpa memperhatikan struktur internal atau implementasi kodenya. Metode ini menguji sistem dengan memberikan berbagai *input* dan memverifikasi keluaran yang dihasilkan sesuai dengan harapan. Pendekatan ini telah banyak diteliti karena efektivitasnya dalam mengidentifikasi *bug* yang muncul akibat kesalahan logika atau interaksi antar modul, tanpa memerlukan akses ke kode sumber.

Menurut Parlika dkk. [30], studi literatur secara sistematis mengenai karakteristik dan tantangan *black-box testing* menunjukkan bahwa pendekatan ini memiliki keunggulan dalam objektivitas pengujian serta kemampuannya untuk mengevaluasi fungsionalitas sistem secara menyeluruh. Dengan demikian, *black-box testing* menjadi komponen penting dalam proses verifikasi untuk memastikan bahwa sistem memenuhi kebutuhan pengguna secara tepat.

2.19. Postman

Postman adalah salah satu alat yang populer dalam pengembangan dan pengujian API. Postman menyediakan antarmuka yang intuitif untuk membuat, mengirim, dan mengelola berbagai skenario pengujian API. Alat ini tidak hanya mempermudah pembuatan dan dokumentasi permintaan HTTP, tetapi juga mendukung otomatisasi pengujian melalui fitur *scripting*, sehingga membantu tim pengembang dalam mendeteksi kesalahan secara dini.

Postman telah digunakan secara luas dalam pengujian API karena kemampuannya untuk menyimulasikan berbagai kondisi permintaan dan respons secara efisien. Hal ini sangat membantu dalam memastikan konsistensi dan keandalan API yang dikembangkan, terutama dalam konteks pengembangan sistem yang bersifat iteratif dan responsif terhadap perubahan kebutuhan.

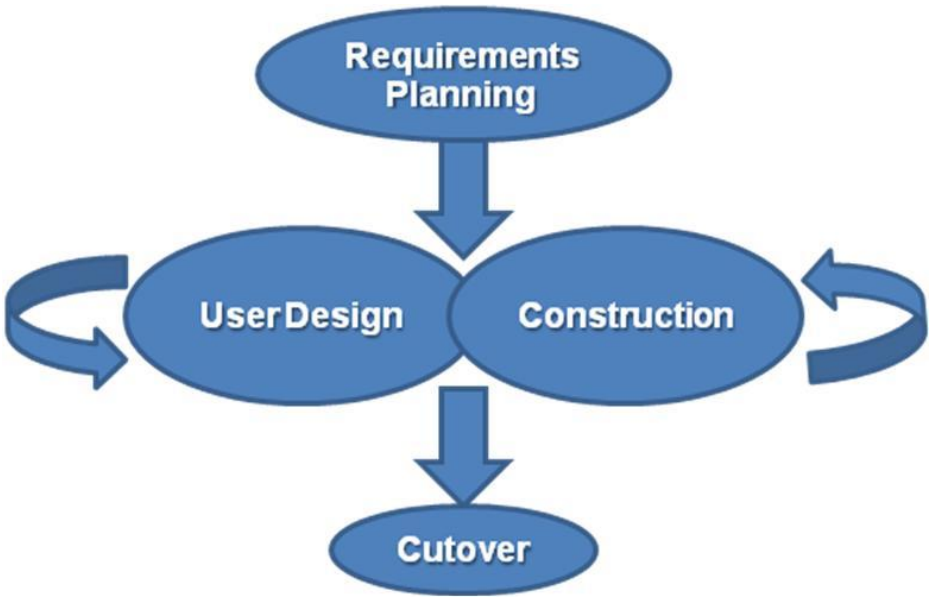
2.20. *Real-Time System*

Konsep real-time data processing didefinisikan sebagai proses penanganan dan analisis data secara langsung pada saat data dihasilkan, berbeda dengan batch processing yang dilakukan secara periodik. Melalui pendekatan ini, data diproses, dianalisis, dan ditindaklanjuti dalam rentang waktu milidetik hingga detik setelah dibuat, sehingga memungkinkan pengambilan keputusan berbasis informasi terkini. Menurut Achanta, keunggulan utama dari pemrosesan waktu-nyata terletak pada kecepatannya dalam menyediakan wawasan segera, akurasinya dalam meminimalkan kesalahan akibat keterlambatan, serta kemampuannya untuk memproses data dalam aliran kontinu dengan skalabilitas tinggi [31].

2.21. *Rapid Application Development* (RAD)

Rapid Application Development (RAD) adalah metode pengembangan perangkat lunak yang berfokus pada kecepatan dan keterlibatan langsung pengguna dalam setiap tahapnya. Tujuan utama dari metode ini adalah menghasilkan sistem yang sesuai dengan kebutuhan pengguna dalam waktu yang lebih singkat dibandingkan metode tradisional [32], [33]. Umumnya, pengembangan sistem memerlukan waktu setidaknya 180 hari, namun dengan metode RAD, proses tersebut dapat dipersingkat menjadi sekitar 60-90 hari [34].

RAD terdiri dari empat fase utama, yaitu Perencanaan Kebutuhan (*Requirements Planning*), Desain Pengguna (*User Design*), Konstruksi (*Construction*), dan *Cutover* [35].



Gambar 2.10 Tahapan Pengembangan dalam Metode RAD

1. Perencanaan Kebutuhan (*Requirements Planning*)

Fase ini menggabungkan elemen dari fase perencanaan sistem dan analisis sistem dalam *System Development Life Cycle* (SDLC). Pada tahap ini, pengguna, manajer, dan tim IT berdiskusi serta menyepakati kebutuhan bisnis, ruang lingkup proyek, batasan, dan persyaratan sistem. Diskusi ini penting agar semua pihak memiliki pemahaman yang jelas sebelum melanjutkan ke tahap berikutnya. Fase ini berakhir ketika tim menyepakati isu-isu utama dan mendapatkan persetujuan manajemen untuk melanjutkan pengembangan.

2. Desain Pengguna (*User Design*)

Pada fase ini, pengguna berinteraksi dengan analis sistem untuk mengembangkan model dan prototipe yang merepresentasikan semua proses, masukan, dan keluaran sistem. RAD menggunakan *Joint Application Design* (JAD) untuk menerjemahkan kebutuhan pengguna menjadi model yang dapat diuji dan dimodifikasi secara interaktif. Proses ini memungkinkan pengguna untuk memahami, mengubah, dan akhirnya menyetujui model kerja sistem sebelum pengembangan lebih lanjut..

3. Konstruksi (*Construction*)

Fase ini berfokus pada tugas pengembangan program dan aplikasi, mirip dengan proses dalam SDLC. Namun, dalam RAD, pengguna tetap terlibat dan dapat memberikan masukan atau perubahan selama pengembangan tampilan, laporan, atau fitur lainnya. Proses dalam fase ini mencakup pemrograman, pengembangan aplikasi, pengodean, pengujian unit, integrasi, dan pengujian sistem secara menyeluruh. Karena RAD menggunakan alat otomatisasi, pengujian dan revisi dapat dilakukan lebih cepat dibandingkan metode tradisional.

4. Cutover

Fase *cutover* mirip dengan tahap implementasi dalam SDLC, yang mencakup konversi data, pengujian sistem akhir, peralihan ke sistem baru, serta pelatihan pengguna. Berbeda dengan metode tradisional, RAD mempercepat seluruh proses ini sehingga sistem baru dapat dikembangkan, dikirimkan, dan mulai digunakan dalam waktu yang lebih singkat. Perencanaan *cutover* harus dimulai sejak tahap awal agar transisi berjalan lancar tanpa mengganggu operasional organisasi.

2.22. *Joint Application Development*

Joint Application Development (JAD) merupakan teknik kolaboratif yang krusial dalam metodologi RAD, terutama pada fase *user design*. Pada fase ini, JAD memfasilitasi diskusi intensif antara pengguna akhir dan tim pengembang untuk merancang antarmuka dan interaksi sistem. Dengan mengadakan sesi terstruktur yang dipimpin oleh fasilitator, *stakeholder* dapat bersama-sama mengidentifikasi kebutuhan, mengevaluasi desain antarmuka, dan menghasilkan prototipe secara cepat.

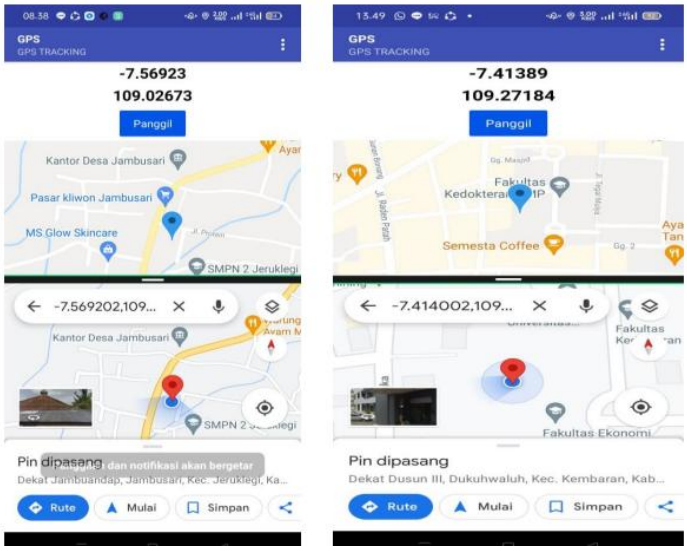
Menurut Martin [35], JAD dalam RAD mempersingkat waktu pengembangan dengan mengurangi iterasi revisi yang panjang. JAD memungkinkan pengguna memberikan masukan langsung tentang tata letak, navigasi, dan elemen visual, yang berdampak pada peningkatan *usability* dan *user experience*. Sesi JAD dalam fase ini juga membantu meminimalkan kesenjangan antara asumsi pengembang dengan kebutuhan nyata pengguna, sehingga desain yang dihasilkan menjadi lebih intuitif dan responsif terhadap kebutuhan pasar.

2.23. Penelitian Terkait

Kesamaan studi kasus dan penggunaan metode dalam pengembangan sistem adalah beberapa hal yang dapat dijadikan referensi untuk penelitian ini. Berikut adalah beberapa artikel atau jurnal ilmiah yang relevan dengan penelitian ini:

1. **Rancang Bangun Alat Bantu Mobilitas Tunanetra dan Penentu Lokasi Menggunakan *Global Positioning System Tracking* Berbasis *Internet Of Things***

   Artikel ini membahas pengembangan alat bantu mobilitas berbasis IoT untuk penyandang tunanetra, yang dirancang untuk mendeteksi halangan, genangan air, dan memberikan informasi lokasi pengguna secara *real-time* kepada keluarga melalui aplikasi yang terhubung dengan Google Maps. Modul GPS Neo Ublox M6 bertugas melacak lokasi pengguna secara *real-time* dan mengirimkan data koordinat ke Firebase melalui koneksi *Wi-Fi*, yang kemudian ditampilkan pada aplikasi Kodular dalam bentuk posisi pengguna di Google Maps. Modul GPS ini memungkinkan fitur *live-tracking* karena data lokasi dikirimkan secara terus-menerus, sehingga keluarga dapat memantau posisi pengguna secara *real-time* [6].



Gambar 2.11 Contoh Aplikasi Google Maps yang Digunakan pada Penelitian

2. **Perancangan Tongkat Bantu Inovatif untuk Tunanetra dengan Memanfaatkan Teknologi Sensor Gyroscope, GPS, dan Ultrasonik**

   Artikel ini membahas pengembangan tongkat bantu inovatif untuk penyandang tunanetra dengan memanfaatkan teknologi sensor giroskop, GPS, dan ultrasonik. Modul GPS Neo 6M digunakan untuk membaca koordinat lokasi yang dapat dikirimkan melalui *bot* Telegram berdasarkan permintaan atau kondisi tertentu, seperti tombol darurat, tetapi tidak disebutkan bahwa sistem ini mendukung pelacakan lokasi secara *real-time*. Hasil pengujian menunjukkan bahwa sistem ini efektif dalam membantu tunanetra dengan mendeteksi halangan, memberikan informasi medan, dan memungkinkan keluarga memantau lokasi pengguna melalui *bot* Telegram. Sayangnya tidak dijelaskan mengenai bagaimana transfer data yang dilakukan antara IoT dengan Telegram [7].

Gambar 2.12 Contoh Pengujian Lokasi Menggunakan *Bot* Telegram

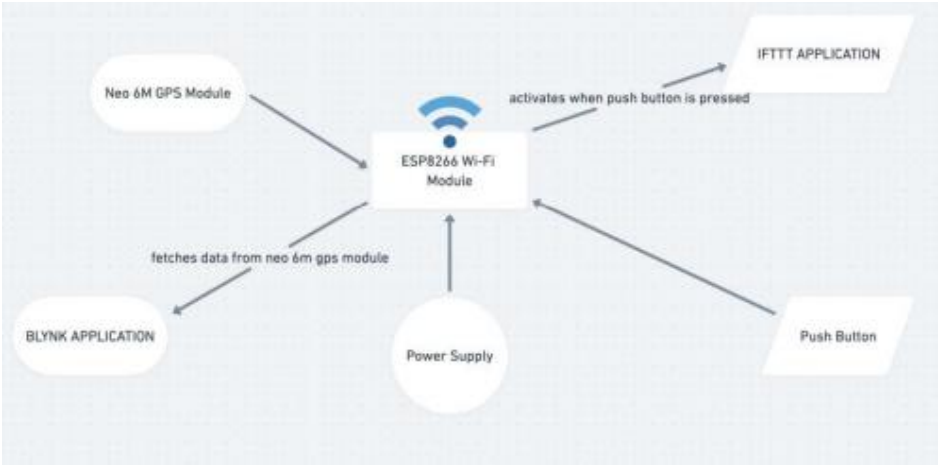
3. **An Intelligent and Multi-Functional Stick for Blind People Using IoT**

   Artikel ini membahas pengembangan tongkat pintar berbasis IoT untuk membantu penyandang tunanetra dalam mendeteksi hambatan, menemukan tongkat yang hilang, dan melacak lokasi pengguna. Sistem ini menggunakan modul GPS untuk membaca koordinat lokasi dan modul GSM untuk mengirimkan informasi lokasi ke keluarga pengguna melalui SMS. Pengiriman lokasi ini dilakukan ketika tombol C pada remot kontrol ditekan, yang mengaktifkan sistem GPS dan GSM untuk mengirimkan koordinat lokasi dalam bentuk pesan teks ke kontak darurat. Pada penelitian ini tidak dijelaskan secara mengenai proses pengiriman data dari modul GPS melalui SMS [8].

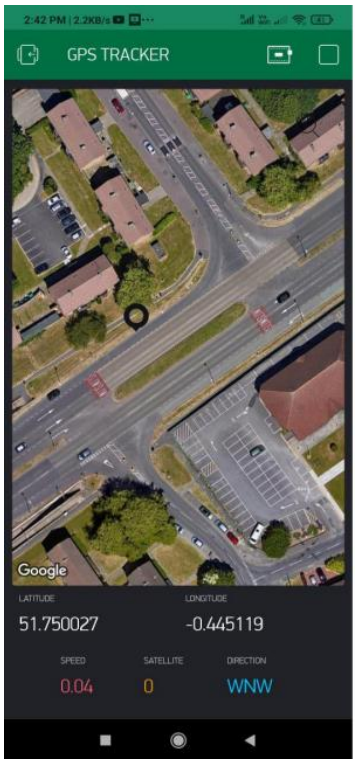
4. **Internet of Things (IoT) Enabled Smart Navigation Aid for Visually Impaired**

   Artikel ini membahas pengembangan alat bantu navigasi pintar berbasis IoT untuk penyandang tunanetra, yang dirancang untuk mendeteksi hambatan, melacak lokasi pengguna, dan menyediakan fitur darurat. Sistem ini menggunakan dua mikrokontroler: Arduino Uno R3 untuk mengelola data dari sensor dan memberikan respons lokal, serta ESP8266 untuk menangani komunikasi berbasis *Wi-Fi*, seperti mengunggah data lokasi ke *cloud* dan memproses notifikasi darurat. Pembagian tugas ini diasumsikan dilakukan untuk memastikan efisiensi pemrosesan dan memanfaatkan kemampuan spesifik dari masing-masing perangkat. Data lokasi diperbarui setiap 20 milidetik melalui modul GPS yang terhubung ke aplikasi berbasis *cloud*

menggunakan ESP8266. Tongkat ini juga memiliki tombol darurat yang, saat ditekan, mengirimkan pesan darurat berisi koordinat lokasi pengguna melalui platform IFTTT kepada kontak darurat. Sistem ini dirancang menggunakan aplikasi Blynk untuk menampilkan pelacakan lokasi secara visual [36].



Gambar 2.13 *Block Diagram* dari Sistem *Location Tracking*



Gambar 2.14 Aplikasi *Location Detection* Menggunakan Blynk

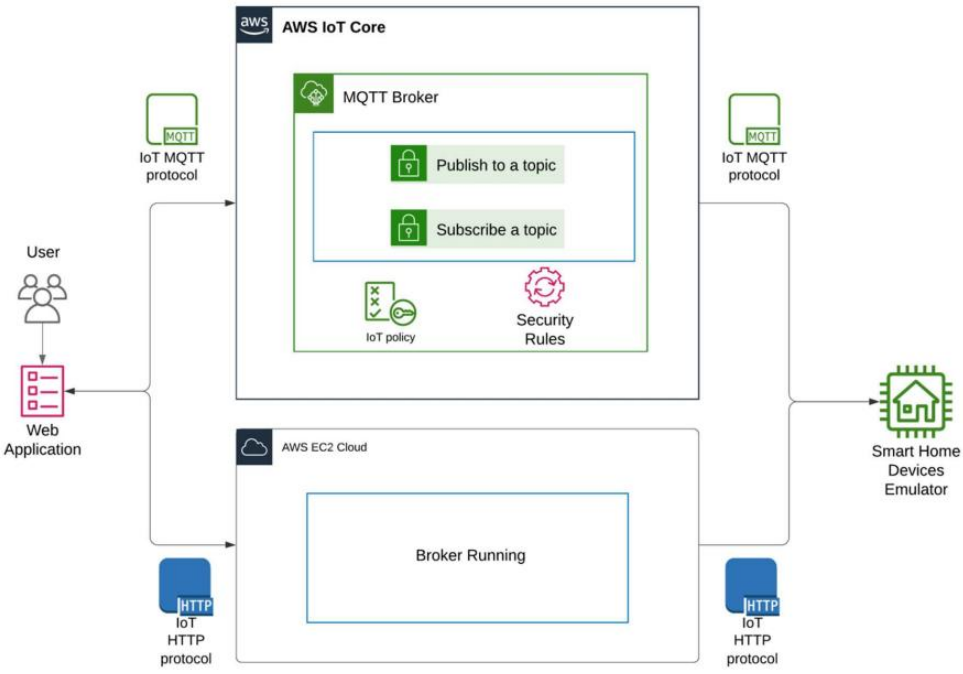
5. **IoT real time data acquisition using MQTT protocol**

   Artikel ini membahas penerapan protokol MQTT untuk akuisisi data *real-time* dalam sistem berbasis IoT. Studi ini mengevaluasi efektivitas MQTT dalam mentransmisikan data sensor dibandingkan dengan protokol HTTP. Sistem yang dikembangkan menggunakan sensor suhu dan kelembaban DHT11 yang terhubung ke mikrokontroler Wemos D1 Mini yang dilengkapi dengan modul *Wi-Fi* ESP8266. Data yang diperoleh dari sensor dikirimkan ke broker MQTT (Mosquitto) sebelum diteruskan ke aplikasi berbasis web dan *mobile* untuk pemantauan secara *real-time*. Protokol MQTT menggunakan model *publish/subscribe*, di mana perangkat yang mengirim data bertindak sebagai *publisher*, sedangkan aplikasi pemantauan bertindak sebagai *subscriber* yang menerima data dari broker. Dibandingkan dengan HTTP, MQTT memiliki keunggulan dalam efisiensi *bandwidth*, latensi yang lebih rendah, dan konsumsi daya yang lebih hemat, menjadikannya lebih cocok untuk aplikasi IoT yang membutuhkan respons cepat dan koneksi stabil. Hasil pengujian menunjukkan bahwa MQTT mampu mentransmisikan data hingga enam kali lebih cepat dibandingkan HTTP dengan tingkat kehilangan data yang lebih rendah, karena arsitektur MQTT yang lebih ringan dan efisien. Implementasi MQTT dalam penelitian ini menunjukkan peningkatan kualitas dan keandalan akuisisi data secara *real-time*, menjadikannya solusi yang lebih optimal untuk sistem pemantauan IoT [10].

6. **Smart Home Application using HTTP and MQTT as Communication Protocols**

   Artikel ini membahas pengembangan sistem rumah pintar yang dapat dikendalikan melalui protokol komunikasi IoT, yaitu HTTP dan MQTT. Sistem ini dikembangkan sebagai respons terhadap kebutuhan interaksi tanpa sentuhan akibat pandemi COVID-19, dengan tujuan mengontrol perangkat seperti lampu pintar, kipas, AC, dan kunci pintu secara jarak jauh melalui aplikasi berbasis web. Implementasi menggunakan dua pendekatan: HTTP berbasis RESTful API yang diterapkan di AWS EC2/Lambda, dan *MQTT-over-WSS* yang diterapkan menggunakan layanan AWS IoT Core. Sistem terdiri dari beberapa komponen utama, yaitu IoT *Emulator App* untuk simulasi perangkat pintar,

Device App yang bertindak sebagai perantara komunikasi antara emulator dan pengguna, serta *Client App* yang digunakan untuk mengontrol perangkat melalui antarmuka pengguna. Protokol MQTT menggunakan model *publish/subscribe* yang memungkinkan komunikasi lebih efisien dibandingkan HTTP, yang berbasis *request-response*. Evaluasi menunjukkan bahwa MQTT lebih unggul dalam efisiensi *bandwidth*, latensi lebih rendah, dan konsumsi daya lebih hemat dibandingkan HTTP, terutama dalam skenario komunikasi *real-time*. Untuk meningkatkan keamanan, HTTP dilengkapi dengan enkripsi RSA dan *hashing* MD5 sebagai sistem autentikasi berbasis tanda tangan digital, sedangkan MQTT menggunakan sistem keamanan bawaan dari AWS IoT. Hasil pengujian menunjukkan bahwa total latensi dalam komunikasi HTTP mencapai 1451 milidetik, sedangkan MQTT memiliki latensi yang lebih rendah, menjadikannya pilihan yang lebih optimal untuk sistem IoT yang membutuhkan respons cepat [37].

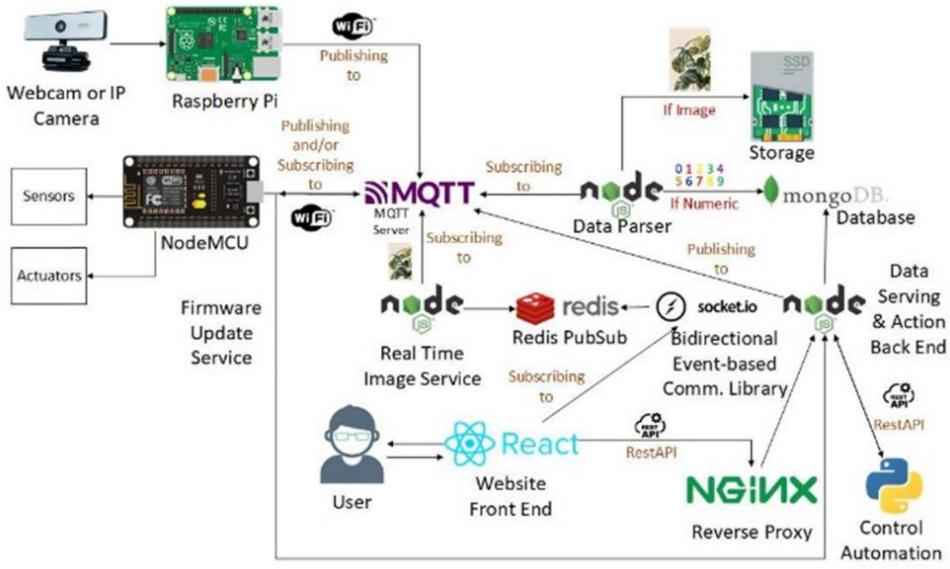


Gambar 2.15 Diagram Arsitektur dari Aplikasi *Smart Home*

7. **Design of Smart Farming Communication and Web Interface Using MQTT and Node.js**

   Artikel ini membahas pengembangan sistem komunikasi dan antarmuka web untuk pertanian pintar menggunakan protokol MQTT dan Node.js. Sistem ini

dirancang untuk meningkatkan efisiensi dalam pertanian presisi dengan mengoptimalkan komunikasi antara perangkat IoT dan server. Data dari sensor dan kamera dikirimkan melalui MQTT ke server, di mana sistem memproses data numerik dan umpan gambar secara langsung. Server menyimpan semua data yang diterima untuk analisis dan penggunaan di masa depan. Antarmuka *backend* dari sistem ini mencakup berbagai fungsi, seperti manajemen *dataset*, manajemen perangkat, administrasi pengguna, manajemen *firmware*, kontrol perangkat, dan pemantauan umpan gambar secara langsung. Node.js digunakan sebagai server sisi *backend* karena kemampuannya dalam menangani permintaan secara asinkron dan efisien. Hasil pengujian menunjukkan bahwa sistem ini dapat menangani hingga 400 pengguna secara bersamaan pada antarmuka web, dengan konsumsi RAM maksimum sebesar 389 MB, sedangkan komunikasi berbasis MQTT mampu menangani hingga 900 perangkat yang terhubung secara simultan. Implementasi ini membuktikan bahwa kombinasi MQTT dan Node.js dapat digunakan untuk mengelola sistem pertanian pintar dengan komunikasi yang efisien dan skalabilitas yang baik [38].



Gambar 2.16 Arsitektur Sistem dan Arus Komunikasi dari Aplikasi

8. Designing A Node.js Full Stack Web

Artikel ini membahas perancangan dan pengembangan aplikasi web *full-stack* berbasis Node.js untuk digunakan oleh penyedia layanan internet dalam

mengelola koneksi pelanggan. Aplikasi ini dirancang agar dapat diakses melalui berbagai perangkat, seperti komputer dan ponsel pintar, dengan menggunakan *framework Bootstrap* untuk memastikan tampilan yang responsif. *Backend* aplikasi dikembangkan menggunakan Node.js, yang memungkinkan server yang aman dan fleksibel dengan ekosistemnya yang luas. Selain itu, API juga dikembangkan dalam proses Node.js yang sama untuk menghubungkan sistem dengan basis data. MariaDB digunakan sebagai basis data yang menyimpan informasi koneksi pelanggan, sementara REST API dikembangkan untuk memungkinkan komunikasi antara basis data dan sistem manajemen koneksi. Dalam implementasi sistem, berbagai aspek keamanan dipertimbangkan, termasuk penggunaan HTTPS, autentikasi pengguna menggunakan Passport.js, serta perlindungan API menggunakan JWT untuk memastikan akses yang aman. Sistem juga mengimplementasikan teknik *server-side rendering* menggunakan *Embedded Javascript* (EJS) untuk meningkatkan performa dan keamanan dengan hanya mengirimkan data yang diperlukan kepada pengguna. Pengujian dilakukan dengan metode *black-box*, menunjukkan bahwa API dapat menangani berbagai fungsi pengelolaan koneksi pelanggan, termasuk perubahan kecepatan koneksi, aktivasi atau deaktivasi layanan, serta pemantauan kondisi jaringan. Sistem ini di-*deploy* ke jaringan intranet perusahaan menggunakan PM2 sebagai manajer proses untuk memastikan aplikasi tetap berjalan stabil. Hasil implementasi menunjukkan bahwa aplikasi ini berfungsi sesuai dengan spesifikasi yang dirancang dan dapat digunakan oleh karyawan untuk mengelola layanan pelanggan dengan lebih efisien [39].

9. **Analisis Performa Load Testing Antara Mysql Dan Nosql Mongodb Pada RestAPI Nodejs Menggunakan Postman**

   Artikel ini membahas analisis performa *load testing* antara MySQL dan MongoDB pada REST API berbasis Node.js menggunakan Postman. Studi ini bertujuan untuk mengevaluasi bagaimana kedua basis data menangani beban kerja yang berbeda dalam konteks aplikasi Node.js, dengan mengukur beberapa parameter utama: *response time* (waktu respons), *throughput* (jumlah permintaan yang diproses per detik), dan *error rate* (tingkat kesalahan dalam permintaan yang dikirimkan ke server). Pengujian dilakukan dengan membuat

dua layanan REST API yang menggunakan basis data berbeda, tetapi tetap dalam satu implementasi aplikasi Node.js yang sama. Data yang digunakan berasal dari NorthwindDB, basis data sampel yang umum digunakan untuk pengujian performa sistem manajemen basis data. *Endpoint* yang diuji mencakup berbagai operasi, seperti INSERT, UPDATE, DELETE, serta *query* kompleks yang melibatkan agregasi dan pencarian dengan kondisi tertentu. Hasil pengujian menunjukkan bahwa MySQL memiliki performa lebih baik dibandingkan dengan MongoDB dalam semua aspek yang diuji, dengan waktu respons lebih cepat, *throughput* lebih tinggi, dan tingkat kesalahan lebih rendah. MySQL lebih unggul dalam menangani *query* kompleks dan transaksi, sementara MongoDB lebih cocok untuk skenario dengan data tidak terstruktur dan kebutuhan fleksibilitas skema. Studi ini memberikan wawasan bagi pengembang dalam memilih basis data yang sesuai dengan kebutuhan aplikasi berbasis Node.js, terutama dalam konteks beban kerja yang tinggi dan skenario transaksi yang membutuhkan keandalan tinggi. [40].

10. **MQTT Broker Performance Comparison between AWS, Microsoft Azure and Google Cloud Platform**

    Artikel ini membahas perbandingan performa broker MQTT yang diimplementasikan di tiga layanan *cloud* utama: AWS, Microsoft Azure, dan Google Cloud Platform (GCP). Studi ini bertujuan untuk mengevaluasi bagaimana ketiga platform menangani *throughput*, latensi, jumlah pesan yang diterima, serta performa dalam berbagai tingkat QoS. Pengujian dilakukan dengan menggunakan MQTTLoader sebagai alat uji beban untuk mengukur efisiensi komunikasi antara *publisher* dan *subscriber* melalui broker MQTT yang berjalan pada setiap layanan *cloud*. Infrastruktur pengujian mencakup virtual *machine* berbasis Linux (Ubuntu 20.04) dengan spesifikasi RAM 8 GB dan CPU 2 *core* yang dikonfigurasi untuk menjalankan layanan MQTT menggunakan broker Mosquitto.

    Hasil pengujian menunjukkan bahwa Azure memiliki keunggulan dalam parameter *Average Throughput Subscriber* di setiap tingkat QoS, serta unggul dalam *Average Throughput Publisher* pada QoS 1 dan 2. GCP unggul dalam *throughput publisher* pada QoS 0 dan dalam menerima jumlah pesan terbanyak

pada QoS 1 dan 2, sedangkan AWS tidak menunjukkan performa terbaik pada metrik utama yang diuji. Dari segi latensi, Azure memiliki latensi paling rendah dalam pengiriman pesan, sedangkan GCP lebih unggul dalam jumlah total pesan yang diterima oleh *subscriber*. Dengan hasil ini, penelitian ini memberikan panduan dalam memilih layanan *cloud* terbaik untuk implementasi broker MQTT, di mana Azure lebih cocok untuk aplikasi dengan *throughput* tinggi dan latensi rendah, sementara GCP lebih optimal untuk kebutuhan jumlah pesan yang diterima dalam jumlah besar [41].

11. **Implementing Authentication and session management in an Angular JS single-page application**

    Artikel ini membahas implementasi mekanisme autentikasi dan manajemen sesi dalam *single-page application* (SPA) berbasis AngularJS. Pendekatan yang digunakan memanfaatkan *token-based authentication*, khususnya dengan JWT, untuk mencapai proses autentikasi yang *stateless* dan skalabel. Saat pengguna berhasil *login* melalui *form* yang mengirimkan kredensial ke server, server memvalidasi data dan menghasilkan JWT yang kemudian disimpan di sisi klien. Token ini dilampirkan pada setiap permintaan HTTP berikutnya untuk mengamankan akses ke sumber daya aplikasi.

    Untuk meningkatkan keamanan, token diatur kedaluwarsa setelah 30 menit tanpa aktivitas, sehingga risiko akses tidak sah dapat ditekan. Selain itu, penelitian menyoroti beberapa kebijakan durasi token lain, mulai dari 15 menit hingga 12 jam. Sesi singkat (15–30 menit) memiliki dampak keamanan sangat tinggi karena pengguna lebih sering diminta *login* ulang, sedangkan sesi lebih panjang (2–12 jam) lebih nyaman namun lebih rentan. Pemilihan kebijakan terbaik bergantung pada kebutuhan aplikasi dan profil risikonya.

Token Expiration Policy	Session Duration	Impact on Security
15 minutes	Short-lived	Very High
30 minutes	Short-lived	High
1 hour	Moderate	Medium
2 hours	Moderate	Low
4 hours	Extended	Low
6 hours	Extended	Medium
12 hours	Extended	High

Gambar 2.17 Perbandingan Antara *Token Expiration Policy* dengan Tingkat Keamanannya

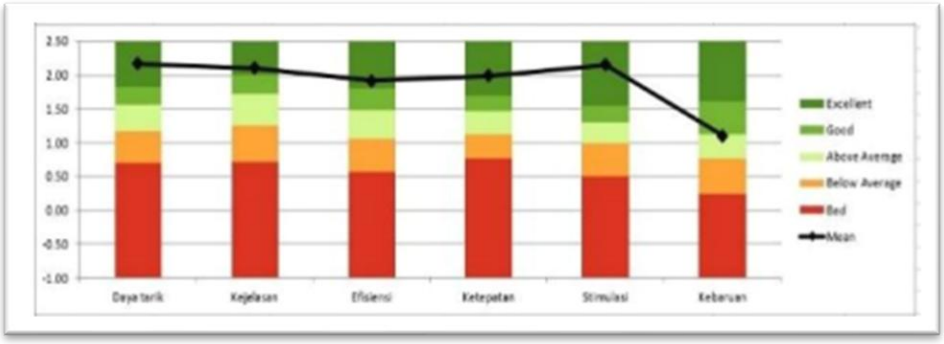
Secara keseluruhan, penelitian ini menunjukkan bahwa dengan menerapkan metode autentikasi dan manajemen sesi yang tepat, AngularJS SPA dapat menyediakan lingkungan aplikasi yang aman, responsif, dan mudah diakses. Penulis juga mengidentifikasi beberapa area perbaikan, seperti peningkatan mekanisme pemulihan kata sandi dan penerapan *multifactor authenticatio*n (MFA), sebagai langkah untuk meningkatkan keamanan sistem lebih lanjut [42].

12. **Pengembangan Sistem Informasi Pendaftaran Seminar Akademik Di Jurusan Teknik Elektro Fakultas Teknik Universitas Lampung Menggunakan Metode Rapid Application Development (RAD)**

    Artikel ini membahas pengembangan sistem informasi pendaftaran seminar akademik di Jurusan Teknik Elektro Universitas Lampung menggunakan metode RAD. Tujuan utama dari penelitian ini adalah untuk menciptakan sistem yang dapat mengelola dan menampilkan data pendaftaran seminar dengan lebih efisien, sehingga informasi dapat diakses dengan cepat dan akurat.

    Sistem yang ada sebelumnya masih menggunakan proses manual, yang menyebabkan berbagai masalah, seperti kurangnya informasi yang terpusat dan kesulitan dalam pencarian data. Dengan penerapan metode RAD, proses pengembangan sistem dilakukan secara bertahap, memungkinkan perbaikan di setiap tahap tanpa harus menyelesaikan seluruh proses perancangan.

Hasil pengujian menunjukkan bahwa sistem baru ini memenuhi kebutuhan pengguna, dengan 40 *use case* yang berhasil diimplementasikan dan diuji. Dari 6 kategori yang dinilai menggunakan metode *User Experience Questionnaire* (UEQ), lima kategori mendapatkan nilai sangat baik, menunjukkan bahwa sistem ini efektif dan efisien dalam mendukung proses pendaftaran seminar akademik. Dengan demikian, pengembangan sistem ini diharapkan dapat meningkatkan kualitas administrasi seminar di lingkungan akademik [32].



Gambar 2.18 Hasil Pengujian UEQ pada Penelitian Terkait

13. Analisis Blackbox Testing Dan User Acceptance Testing Terhadap Sistem Informasi Solusimedsosku

Artikel ini mengevaluasi kinerja dan penerimaan pengguna terhadap sistem informasi Solusimedsosku, sebuah platform *e-commerce* dan SMM panel di Indonesia. Penelitian menggunakan pendekatan *User Acceptance Testing* (UAT) yang diimplementasikan dengan teknik pengujian *black box*. Pendekatan ini menilai sistem secara fungsional, dengan menguji apakah fitur-fitur seperti *login*, registrasi, penambahan pesanan, deposit, dan pengiriman tiket beroperasi sesuai spesifikasi yang ditetapkan tanpa mengakses kode internal.

Selama UAT, pengguna akhir yang berjumlah 45 responden diminta untuk mengisi kuesioner yang menilai sistem dari tiga aspek utama: desain, layanan, dan efisiensi. Hasil pengujian menunjukkan bahwa sistem mendapatkan tingkat penerimaan yang tinggi, dengan persentase rata-rata mencapai 84%—desain 85%, layanan 83%, dan efisiensi 84%. Data ini mengindikasikan bahwa Solusimedsosku telah berhasil memenuhi kebutuhan dan ekspektasi pengguna, serta memberikan pengalaman yang memuaskan.

No	Pertanyaan	Skor				
		1	2	3	4	5
	Desain					
1	Apakah tampilan Solusimedsosku ini menarik?					
2	Apakah menu atau fitur media Solusimedsosku ini mudah dipahami?					
3	Apakah penggunaan warna tulisan dengan latar belakang (*background*) sudah sesuai?					
4	Apakah sistem Solusimedsosku menarik?					
5	Apakah penggunaan tulisan (*font*) mudah dibaca?					
	Layanan					
6	Apakah layanan yang diberikan Solusimedsosku mudah dipahami?					
7	Apakah dengan adanya Solusimedsosku membantu proses dalam penjualan?					
8	Apakah media Solusimedsosku ini dapat dijadikan alat bantu?					
9	Secara keseluruhan apakah penggunaan Solusimedsosku sudah memuaskan?					
10	Apakah Solusimedsosku ini sudah sesuai dengan kebutuhan?					
11	Apakah dengan adanya Solusimedsosku dapat membantu mengurangi beban?					
12	Menurut anda apakah Solusimedsosku ini memiliki kelebihan dibanding aplikasi lainnya?					
	Efisien					
13	Apakah evaluasi lewat Solusimedsosku membantu pengukur pemahaman anda?					
14	Apakah evaluasi pada Solusimedsosku ini sudah sesuai dengan yang dibutuhkan?					
15	Apakah penggunaan Solusimedsosku sudah tepat?					
16	Apakah penggunaan Solusimedsosku ini menghemat biaya opersional?					
17	Apakah penggunaan Solusimedsosku efektif diterapkan dipenjualan?					
18	Apakah penggunaan Solusimedsosku lebih efesien dibandingkan dengan lainnya?					
19	Apakah dengan aplikasi Solusimedsosku ini kostumer lebih memahami?					

Gambar 2.19 Contoh Kuesioner pada Penelitian Terkait

Dengan menggabungkan metode *black-box testing* dalam UAT dan evaluasi melalui kuesioner, penelitian ini menegaskan pentingnya pengujian fungsional dari sudut pandang pengguna untuk memastikan bahwa sistem informasi yang dikembangkan benar-benar siap untuk digunakan di pasar. Hasil yang diperoleh dapat dijadikan acuan dalam pengembangan dan penyempurnaan sistem informasi serupa di masa depan [43].

III. METODE PENELITIAN

3.1.Waktu dan Tempat Penelitian

Pelaksanaan penelitian dan pengembangan alat ini dilakukan pada waktu dan tempat yang telah ditentukan sebagai berikut:

Waktu : Januari 2025 – Juni 2025

Tempat : Laboratorium Teknik Digital Program Studi Teknik Informatika

Adapun rincian waktu penelitian berdasarkan aktivitas yang dilakukan adalah sebagai berikut:

Tabel 3.1 Rincian Waktu Penelitian Berdasarkan Aktivitas

No.	Aktivitas	Januari	Februari	Maret	April	Mei	Juni
1	*Requirements planning*						
2	*User design*						
3	*Construction*						
4	*Cutover*						

3.2.Alat Penelitian

Penelitian ini menggunakan beberapa alat yang mendukung pengembangan sistem *backend* untuk alat bantu tunanetra berbasis IoT. Adapun alat-alat yang digunakan adalah sebagai berikut:

1. Perangkat Keras:
    - Laptop/PC: Digunakan untuk pengembangan *backend* dengan spesifikasi minimum RAM 8 GB, prosesor Intel Core i5 gen 11, dan koneksi internet stabil.
    - Server *Cloud* (Amazon Elastic Compute Cloud)
2. Lingkungan Pengembangan:
    - Sistem operasi yang mendukung pengembangan *backend* berbasis Linux.
    - Jaringan internet stabil untuk menghubungkan perangkat IoT dengan server melalui MQTT.

3.3.Struktur Tim

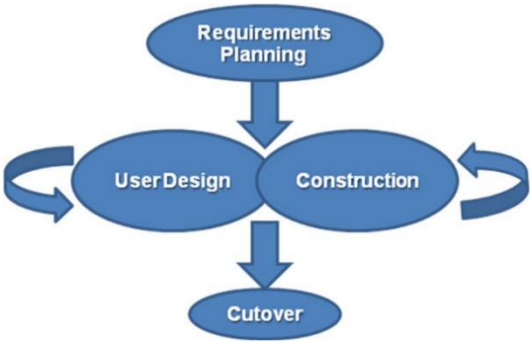
Penelitian ini dilakukan secara tim, dengan masing-masing anggota memiliki peran dan tanggung jawab yang spesifik sesuai dengan bidang keahlian masing-masing. Setiap anggota bertanggung jawab atas pengembangan dan pengujian hasil pekerjaannya masing-masing. Adapun struktur tim dalam penelitian ini adalah sebagai berikut:

Tabel 3.2 Struktur Tim

No.	Nama	Peran	Aplikasi
1.	Pandu Wijaya	*Android Developer*	Teman Jalan
2.	Muhkito Afif	*IoT Developer*	
3.	Alvin Reihansyah Makarim	*Backend Developer*	
4.	Bpk. Chondro	*Product Owner*	

3.4.Tahapan Penelitian

Metode penelitian yang digunakan dalam pengembangan sistem ini adalah RAD. Metode ini dipilih karena memungkinkan pengembangan sistem yang cepat dengan siklus iteratif, di mana pengguna dapat memberikan umpan balik secara langsung untuk perbaikan sebelum implementasi akhir.



Gambar 3.1 Alur metode RAD

3.4.1. *Requirements Planning*

Tahapan perencanaan kebutuhan dilakukan untuk menentukan spesifikasi sistem yang akan dikembangkan. Perencanaan ini mencakup identifikasi kebutuhan pengguna serta kebutuhan sistem secara fungsional dan non-fungsional.

3.4.1.1.Kebutuhan Pengguna

Kebutuhan pengguna mengacu pada kebutuhan utama penyandang tunanetra dan pihak pengawas (keluarga atau pendamping), yaitu:

1. Pengawas membutuhkan informasi lokasi pengguna secara *real-time* untuk memastikan keamanan pengguna saat bepergian.
2. Pengawas mampu menerima data sensor pada aplikasi *mobile*.
3. Sistem harus menyediakan tombol darurat yang dapat mengirimkan notifikasi kepada pengawas saat terjadi situasi kritis, seperti tersesat atau kecelakaan.

3.4.1.2.Kebutuhan Fungsional

Kebutuhan fungsional mencakup fitur-fitur yang harus disediakan oleh sistem agar dapat mendukung kebutuhan pengguna, antara lain:

Tabel 3.3 Kebutuhan Fungsional

ID	Penjelasan
KF-01	Sistem dapat menyediakan layanan pendaftaran bagi pengguna baru.
KF-02	Sistem dapat menyediakan layanan masuk ke aplikasi

Tabel 3.3 Kebutuhan Fungsional (lanjutan)

ID	Penjelasan
KF-03	Sistem dapat menyediakan layanan untuk mengubah kata sandi ketika terlupa
KF-04	Sistem harus dapat menerima data lokasi (*latitude, longitude, timestamp*) dari perangkat IoT melalui MQTT.
KF-05	Sistem harus dapat menerima data sensor dari perangkat IoT, termasuk ketinggian air, dan jarak objek.
KF-06	Sistem harus menyimpan data lokasi dan sensor ke dalam basis data MySQL secara terstruktur.
KF-07	Sistem harus menyediakan *endpoint* API yang memungkinkan aplikasi *mobile* mengambil data lokasi *real-time* pengguna.
KF-08	Sistem harus mendukung pengiriman notifikasi darurat ke aplikasi *mobile* saat tombol *emergency* ditekan.

3.4.1.3.Kebutuhan Non-Fungsional

Kebutuhan non-fungsional menjelaskan persyaratan teknis yang mendukung performa dan keandalan sistem, yaitu:

Tabel 3.4 Kebutuhan Non-Fungsional

ID	Parameter	Penjelasan
KnF-01	*Availability*	Sistem *backend* harus memiliki *uptime* minimal 85%, dapat diakses 24/7, dan responsif terhadap permintaan pengguna.
KnF-02	*Reliability*	Sistem harus tetap stabil meskipun terjadi lonjakan permintaan data dari perangkat IoT dan aplikasi *mobile*.
KnF-03	*Security*	Data lokasi dan informasi pengguna hanya dapat diakses oleh pengguna yang telah Terautentikasi melalui JWT.

Tabel 3.4 Kebutuhan Non-Fungsional (lanjutan)

ID	Parameter	Penjelasan
KnF-04	*Performance*	API *backend* harus merespons permintaan dalam waktu maksimal 5 detik untuk menjaga pengalaman pengguna yang optimal.
KnF-05	*Scalability*	Sistem *backend* harus mampu menangani penambahan jumlah pengguna dan perangkat IoT tanpa menurunkan performa server.
KnF-06	*Maintainability*	*Backend* harus memiliki struktur kode yang modular, sehingga mudah diperbarui atau diperbaiki tanpa mengganggu layanan utama.

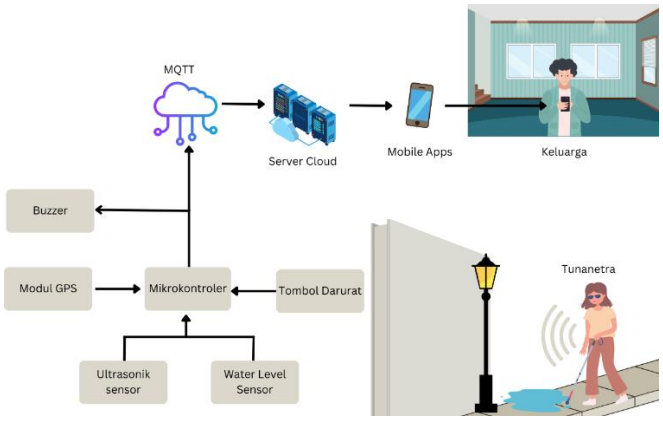
3.4.2. *User Design*

Desain pengguna bertujuan untuk menggambarkan bagaimana komponen dalam sistem bekerja dan berinteraksi. Bagian ini mencakup arsitektur sistem, diagram *use case*, dan diagram relasi entitas untuk menjelaskan struktur dan alur sistem secara keseluruhan.

3.4.2.1.Arsitektur Sistem

A. Arsitektur Sistem Keseluruhan

Arsitektur sistem keseluruhan menggambarkan bagaimana interaksi antara tongkat pintar, server *cloud*, aplikasi seluler, serta pihak keluarga pengguna.



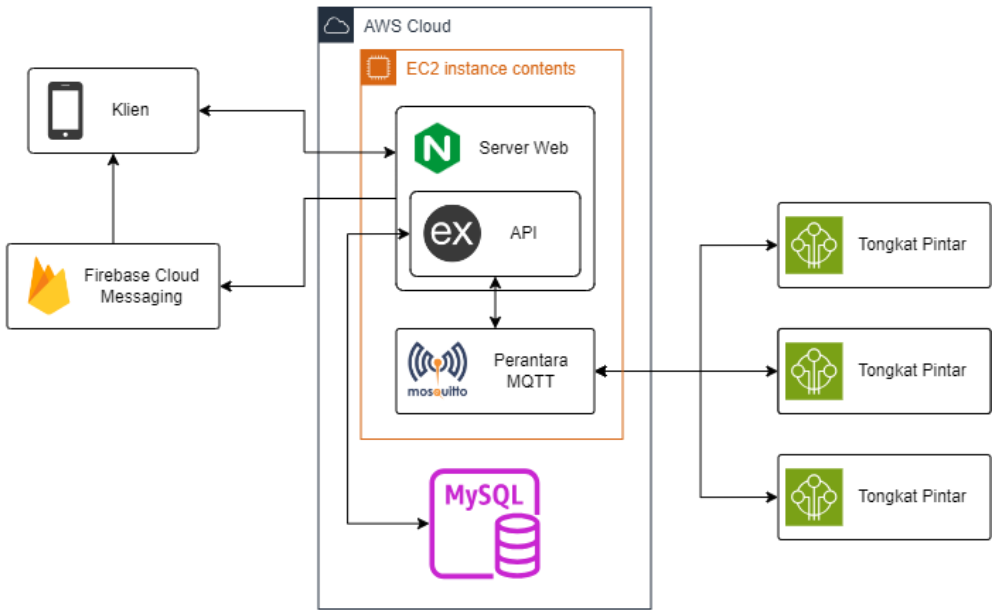
Gambar 3.2 Arsitektur sistem keseluruhan

Tongkat pintar dilengkapi dengan beberapa komponen perangkat keras, antara lain:

1. Sensor ultrasonik untuk mendeteksi rintangan.
2. Sensor *water level* untuk mendeteksi genangan air.
3. Modul GPS untuk melacak lokasi pengguna secara *real-time*.
4. *Buzzer* sebagai alarm peringatan langsung bagi pengguna tunanetra.
5. Tombol darurat untuk mengirimkan sinyal kondisi darurat.
6. Mikrokontroler sebagai pusat kendali perangkat yang mengolah data sensor, lalu mengirimkan informasi ke server menggunakan protokol MQTT.

Data yang dikirimkan dari perangkat *stick* akan diterima oleh aplikasi *backend* di server *cloud*. Server ini kemudian memproses data dan menyimpannya. Aplikasi *backend* akan mengirimkan sesuai dengan permintaan dari aplikasi *mobile*. Dengan demikian, keluarga pengguna dapat memantau kondisi pengguna tongkat pintar dan segera mengambil tindakan jika dibutuhkan.

B. Arsitektur Sistem *Backend*



Gambar 3.3 Arsitektur *backend* sistem

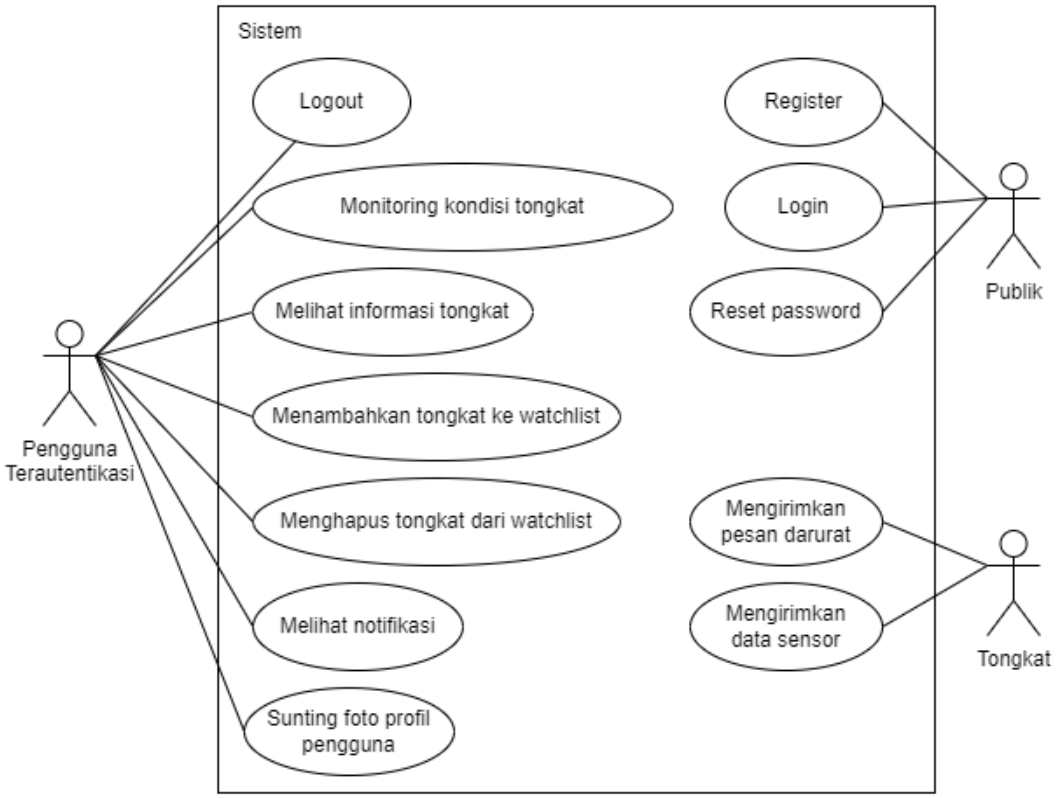
Arsitektur *backend* pengembangan sistem difokuskan untuk membangun fungsi inti aplikasi agar proses komunikasi antara perangkat IoT, server, dan pengguna dapat berjalan dengan baik. Server aplikasi dijalankan pada layanan Amazon EC2. Pada sisi server, digunakan Nginx sebagai *reverse proxy* sekaligus penyaji konten statis

seperti gambar profil pengguna dan gambar tongkat pintar. Aplikasi *backend* dikembangkan dengan Express.js dan dijalankan sebagai aplikasi monolit. *Backend* ini bertanggung jawab dalam mengelola autentikasi, otorisasi, serta transaksi data antara pengguna dengan sistem.

Untuk mendukung komunikasi data secara *real-time* antara perangkat tongkat pintar dan server, digunakan protokol MQTT dengan broker Mosquitto. Broker ini menerima data sensor dari perangkat dan meneruskannya ke aplikasi *backend* untuk diproses lebih lanjut. Seluruh data sistem, termasuk data pengguna dan data tongkat pintar, disimpan dalam basis data MySQL pada Amazon RDS. Selain itu, sistem juga memanfaatkan layanan Firebase Cloud Messaging untuk mengirimkan notifikasi ke aplikasi klien yang terpasang di perangkat pengguna.

3.4.2.2.Diagram *Use Case*

Use case diagram menggambarkan interaksi antara aktor dengan sistem Teman Jalan. Adapun rancangan diagram *use case* pada iterasi pertama adalah sebagai berikut:



Gambar 3.4 Diagram *use case* dari sistem *backend* Teman Jalan

Adapun pendefinisian aktor dalam sistem *backend* adalah sebagai berikut:

Tabel 3.5 Definisi Aktor

No.	Aktor	Deskripsi
1	Publik	Pengguna aplikasi yang belum *login* ke dalam sistem. dan hanya dapat melakukan proses registrasi akun, *login*, serta permintaan reset *password*.
2	Pengguna Terautentikasi	Pengguna aplikasi yang sudah berhasil *login* ke dalam sistem dan dapat melakukan berbagai aktivitas seperti monitoring kondisi tongkat, atau menghapus tongkat dari daftar pantau, dan lain-lain.
3	Tongkat	Tongkat IoT yang digunakan oleh tunanetra.

Adapun pendefinisian *use case* adalah sebagai berikut:

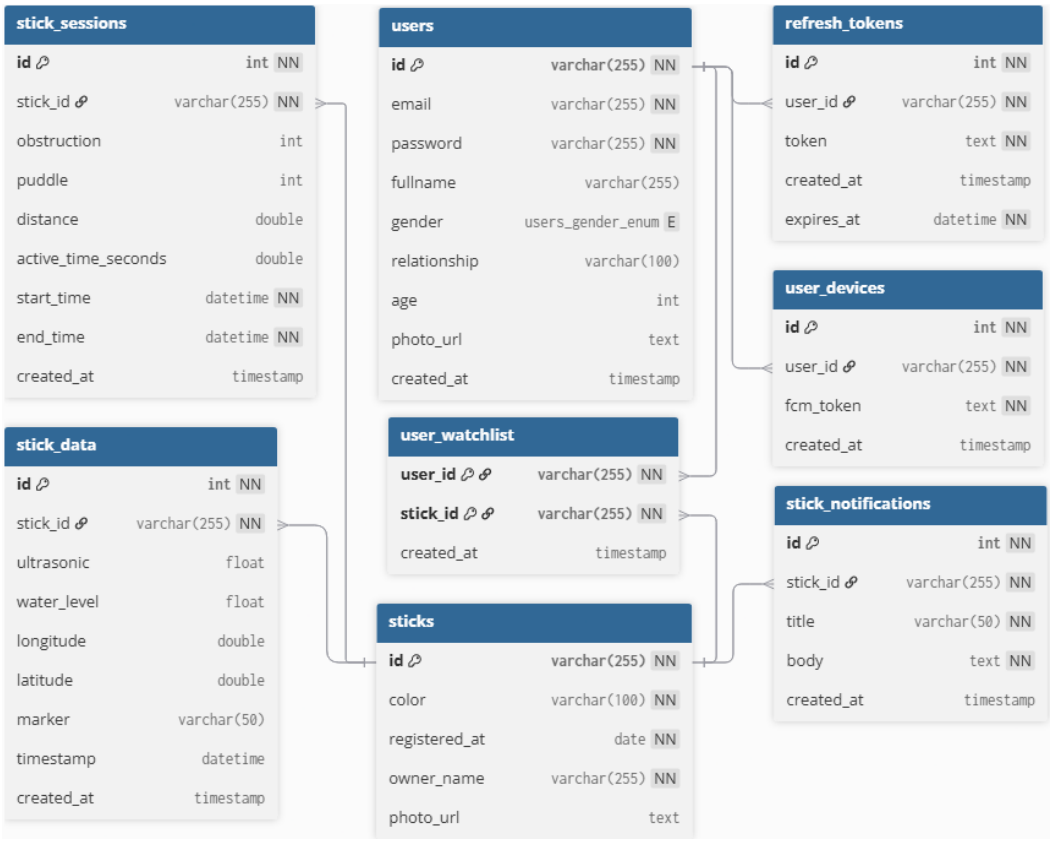
Tabel 3.6 Definisi *use case*

No.	Nama Use Case	Aktor	Deskripsi
1	Register	Publik	Pengguna baru melakukan pendaftaran akun ke sistem.
2	Login	Publik	Pengguna memasukkan email dan *password* untuk masuk ke sistem.
3	Reset Password	Publik	Pengguna yang lupa *password* dapat mengatur ulang dengan OTP.
4	Logout	Pengguna Terautentikasi	Pengguna keluar dari sistem dengan menghapus *refresh token*.
5	Monitoring Kondisi Tongkat	Pengguna Terautentikasi	Pengguna memantau kondisi terkini tongkat pintar (status, statistik).
6	Melihat Informasi Tongkat	Pengguna Terautentikasi	Pengguna melihat detail informasi tongkat tertentu.
7	Menambahkan Tongkat ke Watchlist	Pengguna Terautentikasi	Pengguna menambahkan tongkat ke daftar pantauan.

Tabel 3.6 Definisi *use case* (lanjutan)

No.	Nama Use Case	Aktor	Deskripsi
8	Menghapus Tongkat dari Watchlist	Pengguna Terautentikasi	Pengguna menghapus tongkat dari daftar pantauan.
9	Melihat Notifikasi	Pengguna Terautentikasi	Pengguna melihat daftar notifikasi yang terkait dengan daftar pantau.
10	Menyunting Foto Profil Pengguna	Pengguna Terautentikasi	Pengguna mengganti foto profil pada aplikasi.
11	Mengirimkan Pesan darurat	Tongkat	Pengguna tongkat mengirimkan pesan darurat ke server *backend* dari tongkat.
12	Mengirimkan data sensor	Tongkat	Tongkat mengirimkan data yang diperoleh sensor ke server *backend*

3.4.2.3.Diagram Relasi Entitas



Gambar 3.5 Diagram Relasi Entitas dari Sistem *Backend* Teman Jalan

Entity Relationship Diagram (ERD) menjelaskan bagaimana data pengguna, perangkat *stick*, serta data pendukung lainnya saling berhubungan. Pada aplikasi Teman Jalan, terdapat delapan entitas utama, yaitu users, sticks, stick_data, stick_sessions, stick_notifications, user_watchlist, refresh_tokens, dan user_devices.

Adapun penjelasan mengenai ERD pada sistem ini adalah sebagai berikut:

Tabel 3.7 Entitas pada ERD

No.	Nama Entitas	Deskripsi	Hubungan
1	*Users*	Menyimpan data pengguna aplikasi seperti identitas, kredensial *login*, dan informasi profil.	• 1 user → banyak refresh_tokens • 1 user → banyak user_devices • 1 user → banyak user_watchlist
2	*Sticks*	Merepresentasikan tongkat pintar yang digunakan pengguna.	• 1 stick → banyak stick_data • 1 stick → banyak stick_sessions • 1 stick → banyak stick_notifications • 1 stick ↔ banyak user (via user_watchlist)
3	*Stick Data*	Menyimpan data sensor dari tongkat pintar (ultrasonik, *water level*, lokasi GPS, *marker*).	• Banyak stick_data → 1 stick
4	*Stick Sessions*	Mencatat sesi penggunaan tongkat pintar (jarak tempuh, waktu aktif, start/end time).	• Banyak stick_sessions → 1 stick

Tabel 3.7 Entitas pada ERD (lanjutan)

No.	Nama Entitas	Deskripsi	Hubungan
5	*Stick Notifications*	Menyimpan notifikasi yang dikirimkan tongkat (judul, isi, waktu).	• Banyak stick_notifications → 1 stick
6	*User Watchlist*	Menjadi jembatan *many-to-many* antara *user* dengan tongkat pintar.	• Banyak user ↔ banyak stick
7	*Refresh Tokens*	Menyimpan token autentikasi untuk memperbarui sesi *login* pengguna.	• Banyak refresh_tokens → 1 user
8	*User Devices*	Menyimpan perangkat pengguna (FCM token) untuk pengiriman notifikasi melalui FCM.	• Banyak user_devices → 1 user

3.4.2.4.Diagram Kelas

Class diagram menggambarkan struktur data dan hubungan antar-entitas dalam sistem *backend* yang akan dikembangkan. Setiap kelas mewakili entitas utama yang digunakan untuk menyimpan data dan menyediakan fungsi-fungsi dalam sistem.

Kelas User berfungsi sebagai representasi pengguna yang terdaftar dalam sistem. Setiap pengguna memiliki atribut dasar seperti identitas, email, kata sandi, data profil, serta waktu pembuatan akun. Metode yang dimiliki oleh kelas ini berkaitan dengan proses autentikasi dan pengelolaan akun, meliputi registrasi, *login*, *logout*, pembaruan profil, serta perubahan kata sandi.

Untuk mendukung autentikasi yang aman, digunakan kelas RefreshToken. Kelas ini menyimpan token autentikasi yang memiliki masa berlaku tertentu. Token ini digunakan untuk memperpanjang sesi pengguna tanpa perlu melakukan *login* ulang secara berulang. Kelas ini menyediakan metode untuk menghasilkan, memvalidasi, dan mencabut token.

Selain itu, kelas UserDevice digunakan untuk menyimpan informasi perangkat yang digunakan oleh pengguna, termasuk token FCM yang dipakai dalam

pengiriman notifikasi. Setiap pengguna dapat terhubung dengan lebih dari satu perangkat, sehingga kelas ini berperan dalam manajemen token perangkat.

Kelas UserWatchlist menjadi penghubung antara pengguna dengan tongkat. Melalui kelas ini, pengguna dapat menambahkan atau menghapus tongkat dari daftar pantauan, serta melihat daftar tongkat yang sedang dipantau. Dengan demikian, hubungan antara pengguna dan tongkat bersifat *many-to-many*.

Kelas Stick merepresentasikan tongkat pintar yang menjadi objek utama dalam sistem. Setiap tongkat memiliki atribut seperti identitas, warna, tanggal registrasi, nama pemilik, serta foto. Metode yang tersedia memungkinkan pembaruan informasi tongkat serta pengambilan data detail tongkat.

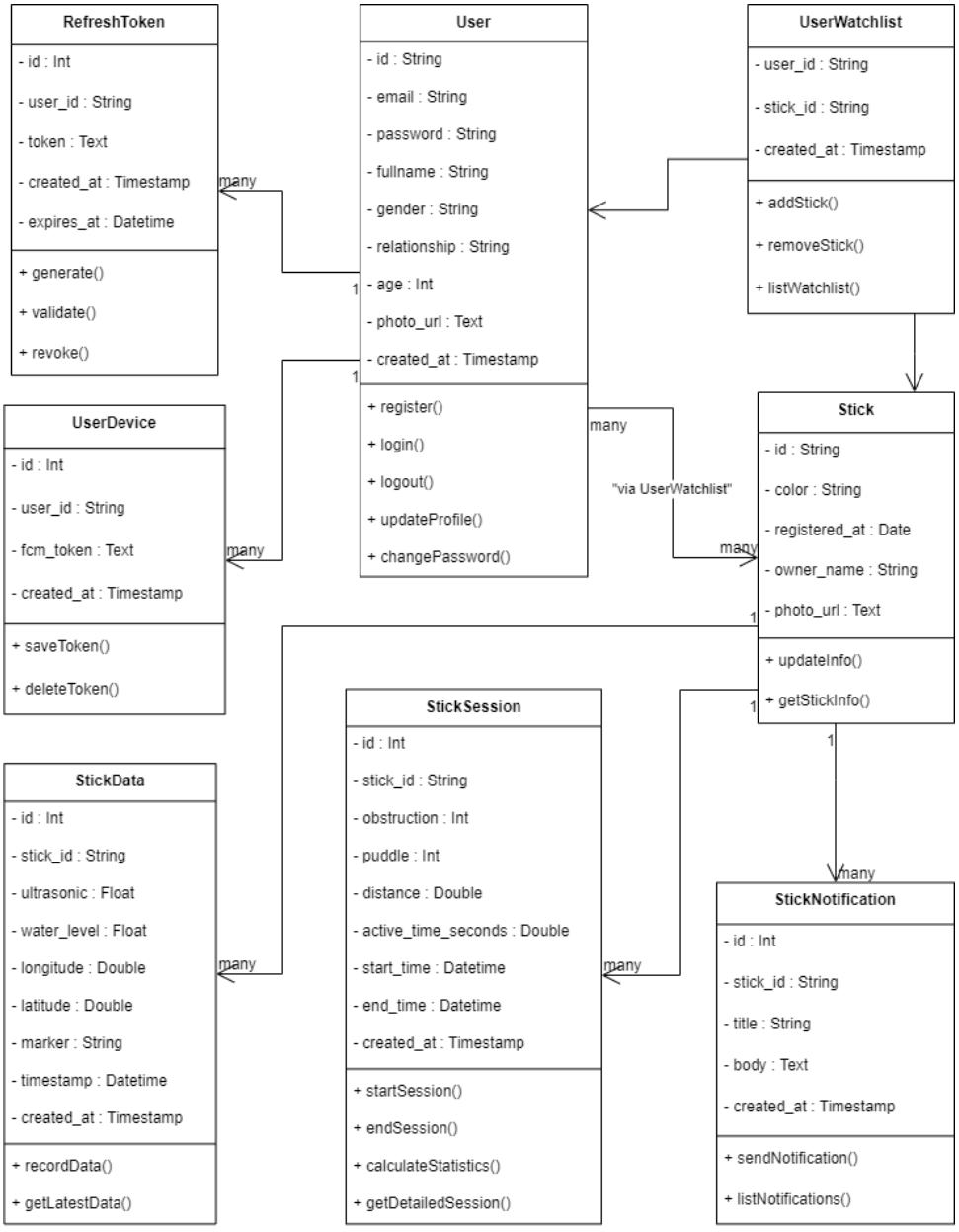
Untuk mencatat penggunaan tongkat, sistem menggunakan kelas StickSession. Kelas ini menyimpan data terkait sesi penggunaan, seperti jumlah hambatan yang terdeteksi, keberadaan genangan air, jarak tempuh, durasi pemakaian, serta waktu mulai dan berakhirnya sesi. Data sesi ini dapat dianalisis lebih lanjut melalui metode perhitungan statistik dan penyajian detail sesi.

Kelas StickData bertugas menyimpan data sensor yang dihasilkan oleh tongkat, misalnya data ultrasonik, ketinggian air, lokasi geografis, serta penanda waktu. Kelas ini memungkinkan pencatatan data sensor baru serta pengambilan data terbaru dari tongkat.

Terakhir, kelas StickNotification merepresentasikan pesan peringatan atau notifikasi yang dikirim dari tongkat kepada pengguna. Setiap notifikasi memiliki judul, isi pesan, serta waktu dibuat. Melalui kelas ini, sistem dapat mengirimkan notifikasi serta menampilkan daftar notifikasi yang pernah dikirim.

Hubungan antara pengguna dan tongkat bersifat *many-to-many* melalui kelas UserWatchlist. Setiap tongkat dapat memiliki banyak sesi penggunaan, banyak data sensor, serta banyak notifikasi. Sementara itu, pengguna dapat memiliki banyak perangkat serta banyak token autentikasi.

Adapun gambar diagram kelas dari sistem ini adalah sebagai berikut:



Gambar 3.6 Diagram Kelas

3.4.3. Construction

Pada fase *construction*, pengembangan sistem *backend* untuk alat bantu tunanetra berbasis IoT akan dilakukan sesuai dengan desain yang telah dirancang sebelumnya. Fase ini akan mencakup pengembangan backend, integrasi dengan perangkat IoT, serta uji coba awal sebelum masuk ke fase *cutover*.

Adapun rencana pelaksanaan fase *construction* adalah sebagai berikut:

1. Pengembangan *Backend*
    - Implementasi API.
    - Integrasi MQTT broker untuk komunikasi dengan perangkat IoT.
    - Pengelolaan data menggunakan MySQL dan Redis.
2. Integrasi dengan Perangkat IoT
    - Menghubungkan perangkat IoT dengan sistem *backend* menggunakan MQTT.
    - Pengujian pengiriman data sensor (lokasi, rintangan, tombol darurat) ke *backend*.
3. Pengujian Black Box Testing
    - Pengujian setiap *endpoint* API menggunakan Postman.
    - Pengujian regresi apabila terdapat perubahan.
    - Pengujian performa untuk memastikan kebutuhan non-fungsional terpenuhi.
4. Pengendalian Versi dan Dokumentasi
    - Penggunaan GitHub untuk manajemen kode dan kolaborasi.
    - Dokumentasi pengembangan, termasuk dokumentasi API dengan Postman dan panduan integrasi perangkat IoT.

Bagian ini bisa diperbarui setelah fase *construction* benar-benar dilakukan untuk mencantumkan hasil-hasil yang dicapai. Dengan demikian, untuk saat ini, bagian ini cukup menjelaskan rencana dan persiapan sebelum implementasi dimulai.

3.4.4. Cutover

Fase *cutover* bertujuan untuk memastikan bahwa sistem *backend* untuk alat bantu tunanetra berbasis IoT siap digunakan oleh pengguna dengan optimal. Adapun tahapan yang akan dilakukan dalam fase ini adalah sebagai berikut:

3.4.4.1.Evaluasi dan Penyempurnaan

Berdasarkan hasil pengujian, sistem akan diperbaiki sebelum digunakan secara penuh. Beberapa tindakan yang mungkin dilakukan:

- Penyempurnaan antarmuka berdasarkan umpan balik pengujian *blackbox*.

- Optimalisasi waktu respons API agar lebih cepat.

3.4.4.2.Dokumentasi dan Pelatihan Pengguna

- Dokumentasi API akan disusun untuk pengembang dan pengguna teknis.
- Jika diperlukan, diberikan pelatihan penggunaan bagi pengguna alat bantu tunanetra dan pengawas mereka.

Fase ini akan dianggap berhasil jika sistem dapat digunakan oleh pengguna tanpa kendala besar, memenuhi kebutuhan fungsional, dan mendapatkan umpan balik positif dari pengguna.

V. KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian, dapat ditarik beberapa kesimpulan sebagai berikut.

1. Penelitian ini berhasil merancang dan membangun sistem *backend* untuk mendukung tongkat pintar IoT bagi penyandang tunanetra sesuai dengan tujuan penelitian. Sistem yang dikembangkan menggunakan framework Node.js yaitu Express.js, MySQL, Redis, dan MQTT serta mampu menyediakan layanan autentikasi, pengelolaan data pengguna, monitoring lokasi real-time, serta pengiriman notifikasi darurat melalui integrasi Firebase Cloud Messaging.
2. Seluruh kebutuhan fungsional yang telah didefinisikan pada tahap perencanaan, seperti registrasi akun, autentikasi dengan JWT, manajemen *watchlist* tongkat, penyimpanan data sensor dan lokasi, serta notifikasi darurat, telah diimplementasikan dan diuji dengan metode *black-box* testing. Hasil pengujian menunjukkan bahwa semua fitur berfungsi sesuai rancangan.
3. Sistem juga memenuhi non-fungsional, di antaranya *availability*, *performance*, *reliability*, *scalability*, *security*, dan *maintainability*. Hal ini dibuktikan melalui pengujian performa menggunakan Postman Performance Test dengan skenario *ramp-up*, *spike*, *peak*, dan *fixed*, yang menunjukkan rata-rata waktu respons < 5 detik dan persentase *error* sangat rendah meskipun diuji dengan 100 *virtual users*.

5.2. Saran

Agar penelitian ini dapat berkembang lebih baik, beberapa saran yang dapat diberikan adalah sebagai berikut:

1. Implementasi sistem keamanan dapat diperkuat melalui penggunaan *multi-factor authentication*, enkripsi data *end-to-end*, serta mekanisme manajemen token yang lebih ketat.
2. Disarankan melakukan pengujian performa dengan skenario pengguna dan perangkat IoT dalam jumlah yang lebih besar, sekaligus mempertimbangkan migrasi ke arsitektur *microservices* untuk mendukung skalabilitas jangka panjang.
3. Sistem dapat diperluas dengan fitur berbasis AI/ML, misalnya untuk mendeteksi pola perjalanan, memperkirakan jarak tempuh, atau memberikan rekomendasi rute aman.

DAFTAR PUSTAKA

[1] M. Imran, "Peningkatan Pemberdayaan Penyandang Tunanetra melalui Perancangan Social Media Newsletter di Yayasan Sosial Tunanetra," *J. Komunitas J. Pengabdi. Kpd. Masy.*, vol. 6, no. 2, hlm. 229–239, Jan 2024, doi: 10.31334/jks.v6i2.3587.

[2] G. K. Anwar, Y. H. Ghani, G. Y. Pa, W. K. Sandjaya, dan T. Handayani, "Alat Bantu Penyandang Tunanetra Emergency Smart Hat," *Aviat. Electron. Inf. Technol. Telecommun. Electr. Controls*, vol. 6, no. 1, hlm. 49, Feb 2024, doi: 10.28989/avitec.v6i1.2025.

[3] M. Wimala dan K. Imanuela, "Perkembangan Internet of Things di Industri Konstruksi," *J. Sustain. Constr.*, vol. 1, no. 2, hlm. 43–51, Mar 2022, doi: 10.26593/josc.v1i2.5701.

[4] J. Manyika *dkk.*, "Unlocking The Potential Of The Internet Of Things," Okt 2017. Diakses: 8 Februari 2025. [Daring]. Tersedia pada: https://aegex.com/images/uploads/white_papers/Unlocking_the_potential_of_the_Internet_of_Things___McKinsey__Company.pdf

[5] United States Coast Guard Navigation Center, "GPS Frequently Asked Questions." Diakses: 10 Februari 2025. [Daring]. Tersedia pada: https://www.navcen.uscg.gov/gps-frequently-asked-questions#1

[6] R. P. Anggara dan A. J. Taufiq, "Rancang Bangun Alat Bantu Mobilitas Tunanetra Dan Penentu Lokasi Menggunakan Global Positioning System Tracking Berbasis Internet Of Things," *J. Ris. Rekayasa Elektro*, vol. 3, no. 2, Des 2021, doi: 10.30595/jrre.v3i2.11627.

[7] A. Pambudi dan M. D. Leonardo, "Perancangan Tongkat Bantu Inovatif Untuk Tunanetra dengan Memanfaatkan Teknologi Sensor Gyroscope, GPS, dan Ultrasonik," *Indones. J. Bus. Intell.*, vol. 6, no. 1, hlm. 63–70, Jun 2023, doi: 10.21927/ijubi.v6i1.3304.

[8] S. Ahmed *dkk.*, "An Intelligent and Multi-Functional Stick for Blind People Using IoT," dalam *2022 3rd International Conference on Intelligent Engineering and Management (ICIEM)*, London, United Kingdom: IEEE, Apr 2022, hlm. 326–331. doi: 10.1109/ICIEM54221.2022.9853012.

[9] I. R. Nugraha, W. H. N. Putra, dan E. Setiawan, "A Comparative Study of HTTP and MQTT for IOT Applications in Hydroponics," *Rekayasa Sist. Dan*

Teknol. Inf., vol. 8, no. 1, hlm. 119–126, Feb 2024, doi: 10.29207/resti.v8i1.5561.

[10] R. A. Atmoko, R. Riantini, dan M. K. Hasin, "IoT real time data acquisition using MQTT protocol," *J. Phys. Conf. Ser.*, vol. 853, hlm. 012003, Mei 2017, doi: 10.1088/1742-6596/853/1/012003.

[11] Eclipse Foundation, "Mosquitto - Open Source MQTT Broker." Diakses: 8 Februari 2025. [Daring]. Tersedia pada: https://mosquitto.org/

[12] B. Mishra, B. Mishra, dan A. Kertesz, "Stress-Testing MQTT Brokers: A Comparative Analysis of Performance Measurements," *Energies*, vol. 14, no. 18, hlm. 5817, Sep 2021, doi: 10.3390/en14185817.

[13] E. Bertrand-Martinez, P. Dias Feio, V. D. Brito Nascimento, F. Kon, dan A. Abelém, "Classification and evaluation of IoT brokers: A methodology," *Int. J. Netw. Manag.*, vol. 31, no. 3, hlm. e2115, Mei 2021, doi: 10.1002/nem.2115.

[14] B. Mishra, "Performance Evaluation of MQTT Broker Servers," dalam *Computational Science and Its Applications – ICCSA 2018*, vol. 10963, O. Gervasi, B. Murgante, S. Misra, E. Stankova, C. M. Torre, A. M. A. C. Rocha, D. Taniar, B. O. Apduhan, E. Tarantino, dan Y. Ryu, Ed., dalam Lecture Notes in Computer Science, vol. 10963. , Cham: Springer International Publishing, 2018, hlm. 599–609. doi: 10.1007/978-3-319-95171-3_47.

[15] J. Bloch, "How to design a good API and why it matters," dalam *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, Portland Oregon USA: ACM, Okt 2006, hlm. 506–507. doi: 10.1145/1176617.1176622.

[16] Amazon Web Services, "What is a RESTful API?" Diakses: 10 Februari 2025. [Daring]. Tersedia pada: https://aws.amazon.com/what-is/restful-api/

[17] P. Gowda dan A. N. Gowda, "Best Practices in REST API Design for Enhanced Scalability and Security," *J. Artif. Intell. Mach. Learn. Data Sci.*, vol. 2, no. 1, hlm. 827–830, Feb 2024, doi: 10.51219/JAIMLD/priyanka-gowda/202.

[18] G. Jadhav dan F. Gonsalves, "Role of Node.js in Modern Web Application Development," *Int. Res. J. Eng. Technol.*, vol. 7, no. 6, hlm. 6145–6150, Jun 2020.

[19] A. Amarulloh, K. Kurniasih, dan M. Muchlis, "Analisis Perbandingan Performa Web Service Rest Menggunakan Framework Laravel, Django, Dan Node.Js Pada Aplikasi Berbasis Website," *J. Tek. Inform.*, vol. 9, no. 1, hlm. 14–19, Feb 2023, doi: 10.51998/jti.v9i1.515.

[20] Google Firebase, “Topic messaging on Android.” Diakses: 12 Maret 2025. [Daring]. Tersedia pada: https://firebase.google.com/docs/cloud-messaging/android/topic-messaging

[21] S. Suryawanshi, “Securing the Modern Web: A Comprehensive Exploration of Web API Authentication and Future Trends,” 31 Januari 2025, *Preprints*. doi: 10.36227/techrxiv.173835579.93447825/v1.

[22] K. Sahatqija, J. Ajdari, X. Zenuni, B. Raufi, dan F. Ismaili, “Comparison between relational and NOSQL databases,” dalam *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Opatija: IEEE, Mei 2018, hlm. 0216–0221. doi: 10.23919/MIPRO.2018.8400041.

[23] I. Šušter dan T. Ranisavljević, “Optimization of MySQL database,” *J. Process Manag. New Technol.*, vol. 11, no. 1–2, hlm. 141–151, 2023, doi: 10.5937/jouproman2301141Q.

[24] D. R. Prasetyo, F. Fatoni, M. Nasir, dan I. Effendy, “Implementasi Redis Untuk Mempercepat Pengambilan Data (Studi Kasus: Sistem Dasawisma PKK Sumsel),” *Explore J. Sist. Inf. Dan Telematika*, vol. 15, no. 2, hlm. 214–223, Des 2024, doi: 10.36448/jsit.v15i2.3948.

[25] N. M. A. E. D. Wirastuti, L. Verlin, I.-H. Mkwawa, dan K. G. Samarah, “Implementation of Geographic Information System Based on Google Maps API to Map Waste Collection Point Using the Haversine Formula Method,” *J. Ilm. Tek. Elektro Komput. Dan Inform.*, vol. 9, no. 3, hlm. 731–745, Agu 2023, doi: 10.26555/jiteki.v9i3.26588.

[26] K. Saputra, N. Nazaruddin, D. H. Yunardi, dan R. Andriyani, “Implementation of Haversine Formula on Location Based Mobile Application in Syiah Kuala University,” dalam *2019 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)*, Banda Aceh, Indonesia: IEEE, Agu 2019, hlm. 40–45. doi: 10.1109/CYBERNETICSCOM.2019.8875686.

[27] A. R. Maulana dan A. Rahmatulloh, “Websocket untuk Optimasi Kecepatan Data Transfer pada Real Time Chatting,” *Innov. Res. Inform. Innov.*, vol. 1, no. 1, Mar 2019, doi: 10.37058/innovatics.v1i1.667.

[28] Q. Zhang, L. Cheng, dan R. Boutaba, “Cloud computing: state-of-the-art and research challenges,” *J. Internet Serv. Appl.*, vol. 1, no. 1, hlm. 7–18, Mei 2010, doi: 10.1007/s13174-010-0007-6.

[29] A. Decan, T. Mens, P. R. Mazrae, dan M. Golzadeh, “On the Use of GitHub Actions in Software Development Repositories,” dalam *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Limassol, Cyprus: IEEE, Okt 2022, hlm. 235–245. doi: 10.1109/ICSME55016.2022.00029.

[30] R. Parlika, T. A. Nisaa', S. M. Ningrum, dan B. A. Haque, "Studi Literatur Kekurangan dan Kelebihan Pengujian Black Box," *J. Teknol. Dan Inform.*, vol. 10, no. 2, hlm. 131–140, Sep 2020.

[31] M. Achanta, "The Impact of Real - Time Data Processing on Business Decision - making," *Int. J. Sci. Res. IJSR*, vol. 13, no. 7, hlm. 400–404, Jul 2024, doi: 10.21275/SR24708033511.

[32] H. D. Septama, Y. Mulyani, M. Pratama, dan N. H. Ardike, "Pengembangan Sistem Informasi Pendaftaran Seminar Akademik Di Jurusan Teknik Elektro Fakultas Teknik Universitas Lampung Menggunakan Metode Rapid Application Development (RAD)," *Barometer*, vol. 5, no. 1, hlm. 239–244, Sep 2020, doi: 10.35261/barometer.v5i1.2252.

[33] N. W. Rhomadhona, M. A. Muhammad, P. B. Wintoro, dan Y. Mulyani, "Penerapan Metode Rapid Application Development Untuk Sistem Informasi Event Berbasis Web Pada Universitas Lampung," *J. Inform. Dan Tek. Elektro Terap.*, vol. 13, no. 1, Jan 2025, doi: 10.23960/jitet.v13i1.5597.

[34] S. Aswati, M. S. Ramadhan, A. U. Firmansyah, dan K. Anwar, "Studi Analisis Model Rapid Application Development Dalam Pengembangan Sistem Informasi," *J. Matrik*, vol. 16, no. 2, hlm. 20, Jul 2017, doi: 10.30812/matrik.v16i2.10.

[35] J. Martin, *Rapid application development*. New York : Toronto : New York: Macmillan Pub. Co. ; Collier Macmillan Canada ; Maxwell Macmillan International, 1991. [Daring]. Tersedia pada: https://archive.org/details/rapidapplication00mart

[36] M. Roy dan P. Shah, "Internet of Things (IoT) Enabled Smart Navigation Aid for Visually Impaired," dalam *Advanced Information Networking and Applications*, vol. 451, L. Barolli, F. Hussain, dan T. Enokido, Ed., dalam Lecture Notes in Networks and Systems, vol. 451. , Cham: Springer International Publishing, 2022, hlm. 232–244. doi: 10.1007/978-3-030-99619-2_23.

[37] M. Ahmed dan M. M. Akhtar, "Smart Home: Application using HTTP and MQTT as Communication Protocols," *Zenodo*, Nov 2021, doi: 10.48550/ARXIV.2112.10339.

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

[39] J. Kinnunen, "Designing A Node.js Full Stack Web Application," Bachelor's Thesis, Metropolia University of Applied Sciences, 2023. [Daring]. Tersedia pada:

https://www.theseus.fi/bitstream/handle/10024/793330/Kinnunen_Janne.pdf?sequence=2

[40] L. D. Andrianto dan D. F. Suyatno, “Analisis Performa Load Testing Antara Mysql Dan Nosql Mongodb Pada RestAPI Nodejs Menggunakan Postman,” *J. Emerg. Inf. Syst. Bus. Intell.*, vol. 5, no. 1, hlm. 18–26, Jan 2024.

[41] A. S. S. Ansyah *dkk.*, “MQTT Broker Performance Comparison between AWS, Microsoft Azure and Google Cloud Platform,” dalam *2023 International Conference on Recent Trends in Electronics and Communication (ICRTEC)*, Mysore, India: IEEE, Feb 2023, hlm. 1–6. doi: 10.1109/ICRTEC56977.2023.10111870.

[42] P. G. A. N. Gowda, “Implementing Authentication and session management in an Angular JS single-page application,” *Eur. J. Adv. Eng. Technol.*, vol. 9, no. 7, hlm. 81–86, Jul 2022, doi: 10.5281/ZENODO.13623026.

[43] I. Wahyudi, Fahrullah, F. Alameka, dan Haerullah, “Analisis Blackbox Testing Dan User Acceptance Testing Terhadap Sistem Informasi Solusimedsosku,” *J. Teknosains Kodepena*, vol. 4, no. 1, hlm. 1–9, Agu 2023, doi: 10.54423/jtk.v4i1.54.