PENGEMBANGAN DAN IMPLEMENTASI *BACKEND* & REST-API *DASHBOARD IOT* PADA *INTELLIGENCE CONTROL SYSTEM* (*ICS*) *SMART GREENHOUSE* MENGGUNAKAN PROTOKOL *MQTT*

SKRIPSI

Oleh

M. Affan Siddiqie Asmara

2015031048

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS LAMPUNG

2024

ABSTRAK

PENGEMBANGAN DAN IMPLEMENTASI *BACKEND* & REST-API *DASHBOARD IOT* PADA *INTELLIGENCE CONTROL SYSTEM* (*ICS*) *SMART GREENHOUSE* MENGGUNAKAN PROTOKOL *MQTT*

Oleh

M. AFFAN SIDDIQIE ASMARA

Smart Greenhouse merupakan sistem rumah kaca yang menggunakan teknologi dan IoT dalam mengoptimalkan kondisi pertumbuhan tanaman, *Intelligence control system* mencakup berbagai sensor yang akan melakukan pemantauan terhadap parameter lingkungan, aktuator, dan *climate* daalam *greenhouse*. Penelitian ini bertujuan untuk mengembangkan dan mengimplementasikan *backend dashboard* IoT *intelligence control system* (ICS) di *smart greenhouse* menggunakan protokol *MQTT*. Penelitian ini menggunakan metode pengembangan sistem dengan pendekatan *Software Development Life Cycle (SDLC) prototype*. Tahap pengujian dilakukan melalui pengujian performa *REST-API* yang dirancang untuk memastikan fungsionalitas dan kinerja yang optimal. Hasil pengujian menunjukkan bahwa API yang dikembangkan, terutama pada *endpoint* dengan frekuensi permintaan tinggi, memiliki *respon time* rata-rata 286 ms dan *error rate* 0,01%, yang memenuhi harapan kinerja sistem. Selain itu, hasil pengujian *throughput* menunjukkan bahwa API mampu menangani beban permintaan dengan baik, meskipun terdapat kompleksitas pada beberapa *endpoint* yang mempengaruhi kinerja. Berdasarkan hasil penelitian, API yang dikembangkan dapat memnuhi harapan dan sesuai untuk digunakan dalam pengembangan *dashboard monitoring AGRI-ICS.*

Kata kunci : *Backend, MQTT, VPS, Database, Rest API, Software Development Life Cycle (SDLC)*

ABSTRACT

DEVELOPMENT AND IMPLEMENTATION OF IOT BACKEND & REST-API DASHBOARD ON INTELLIGENCE CONTROL SYSTEM (ICS) SMART GREENHOUSE USING MQTT PROTOCOL

By

M. AFFAN SIDDIQIE ASMARA

Smart Greenhouse is a greenhouse system that uses technology and IoT in optimizing plant growth conditions, Intelligence control system includes various sensors that will monitor environmental parameters, actuators, and climate in the greenhouse. This research aims to develop and implement an IoT intelligence control system (ICS) dashboard backend in a smart greenhouse using the MQTT protocol. This research uses a system development method with a prototype Software Development Life Cycle (SDLC) approach. The testing phase is carried out through REST-API performance testing designed to ensure optimal functionality and performance. The test results show that the developed API, especially on endpoints with high frequency of requests, has an average response time of 286 ms and an error rate of 0.01%, which meets system performance expectations. In addition, the throughput test results show that the API is able to handle the request load well, although there are complexities in some endpoints that affect performance. Based on the research results, the API developed can meet expectations and is suitable for use in the development of the AGRI-ICS monitoring dashboard.

Keywords : *Backend, MQTT, VPS, Database, Rest API, Software Development Life Cycle (SDLC)*

PENGEMBANGAN DAN IMPLEMENTASI *BACKEND* & REST-API *DASHBOARD IOT* PADA *INTELLIGENCE CONTROL SYSTEM* (*ICS*) *SMART GREENHOUSE* MENGGUNAKAN PROTOKOL *MQTT*

Oleh

M. Affan Siddiqie Asmara

SKRIPSI

Sebagai Salah Satu Syarat untuk Mencapai Gelar

SARJANA TEKNIK

Pada

Jurusan Teknik Elektro

Fakultas Teknik Universitas Lampung

FAKULTAS TEKNIK

UNIVERSITAS LAMPUNG

BANDAR LAMPUNG

2024

Judul Skripsi	:	**PENGEMBANGAN DAN IMPLEMENTASI *BACKEND & REST-API DASHBOARD IOT* PADA *INTELLIGENCE CONTROL SYSTEM (ICS) SMART GREENHOUSE* MENGGUNAKAN PROTOKOL MQTT**
Nama Mahasiswa	:	*M. Affan Siddiqie Asmara*
Nomor Pokok Mahasiswa	:	**2015031048**
Jurusan	:	**Teknik Elektro**
Fakultas	:	**Teknik**

MENYETUJUI

1. Komisi Pembimbing

Pembimbing Pendamping	Pembimbing Utama
Dr. Ing. Ardian Ulvan, S.T., M.Sc.	**Yetti Yuniati, S.T., M.T.**
NIP. 197311281999031005	NIP. 198001132009122002

2. Mengetahui

Ketua Jurusan Teknik Elektro	Ketua Program Studi Teknik Elektro
Herlinawati, S.T.,M.T.	**Sumadi, S.T., M.T.**
NIP. 197103141999032001	NIP. 197311042000031001

MENGESAHKAN

1. Tim Penguji

   Ketua : **Yetti Yuniati, S.T., M.T.** .....

   Sekretaris : **Dr. Ing. Ardian Ulvan, S.T., M.Sc.** .....

   Penguji : **Dr. Ing. Melvi, S.T., M.T.** .....

2. Dekan Fakultas Teknik

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

NIP. 197509282001121002

Tanggal Lulus Ujian Skripsi : **9 Agustus 2024**

SURAT PERNYATAAN

Dengan ini saya menyatakan bahwa skripsi ini tidak terdapat karya yang pernah dilakukan orang lain dan sepanjang sepengetahuan saya tidak terdapat atau diterbitkan oleh orang lain, kecuali secara tertulis diacu dalam naskah ini sebagaimana yang disebutkan dalam daftar pustaka. Selain itu, saya menyatakan pula bahwa skripsi ini dibuat oleh saya sendiri.

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

Bandar Lampung, 7 September 2024

M. Affan Siddiqie Asmara
NPM. 2015031048

RIWAYAT HIDUP

Penulis Lahir di Padang pada tanggal 12 April 2002 sebagai anak kedua dari tiga bersaudara dari pasangan Bapak Asmaranur dan Ibu Aniyar. Penulis memulai pendidikan di TKIT AN-Nizam Medan pada tahun 2007, SDIT AN-Nizam pada tahun 2007 hingga 2009, SDN 07 Nan Sabaris pada tahun 2009 hingga 2014, MTsN 2 Padang Pariaman pada tahun 2014 hingga 2017 dan SMAN 1 Nan Sabaris pada tahun 2017 hingga 2020.

Penulis menjadi mahasiswa Jurusan Teknik Elektro, Universitas Lampung pada tahun 2020 melalui jalur SBMPTN (Seleksi Bersama Masuk Perguruan Tinggi Negeri). Selama menjadi mahasiswa , penulis aktif berorganisasi pada Himpunan Mahasiswa Teknik Elektro (HIMATRO) Universitas Lampung sebagai Anggota Divisi Pendidikan pada periode 2021 dan Divisi Humas pada 2022. Penulis juga berkesempatan menjadi aisten praktikum mata kuliah Sistem Komunikasi pada tahun 2023. Penulis melakukan kerja praktik di PT. Sinergi Transformasi Digital dengan *project Cloud Computing Framework for Business* dengan mengangkat judul laporan "Perencanaan Migrasi *On-Premise Server* Menjadi Layanan Berbasis *Cloud Services* Menggunakan CSP *Microsoft Azure*".

PERSEMBAHAN

Alhamdulillah, Atas Izin Allah yang Maha Kuasa

KUPERSEMBAHKAN KARYA INI UNTUK

Ayah dan Ibu Tercinta

ASMARANUR

&

ANIYAR

Dan Saudara – Sauradaku Tersayang

ADZKIA CELINA ASMARA

ASHILA PERMATA ASMARA

KELUARGA BESAR, DOSEN, TEMAN DAN

ALMAMATER

MOTTO

“Allah tidak membebani seseorang melainkan sesuai dengan kesanggupannya”.
(Q.S Al-Baqarah:286)

"Tahapan pertama dalam mencari ilmu adalah mendengarkan, kemudian diam dan menyimak dengan penuh perhatian, lalu menjaganya, lalu mengamalkannya, dan kemudian menyebarkannya".
(Sufyan bin Uyainah)

“*No man can win every battle, but no man should fall without a struggle*”.
(Peter Parker)

"Semua manusia punya batas. Mereka belajar apa adanya dan mereka belajar untuk tidak melampauinya. Saya mengabaikan itu".
(Chuck Dixon)

*“*Kalau kamu menjanjikan sesuatu, harus kamu tepati, kalau kamu membuat kesalahan, kamu harus minta maaf, Dan kalau kamu memberi seseorang mimpi, kamu harus menjaga mimpinya sampai akhir*“*.
(Childe/Tartaglia)

“Aku tidak khawatir akan jadi apa aku di masa depan nanti, apa aku akan berhasil atau gagal. Tapi yang pasti apa yang aku lakukan sekarang akan membentukku di masa depan nanti”.
(Uzumaki Naruto)

SANWACANA

Segala puji bagi Allah, atas limpahan nikmat-Nya yang diberikan kepada penulis sehingga dapat menyelesaikan Tugas Akhir ini. Shalawat dan salam senantiasa dicurahkan kepada Nabi Muhammad, suri teladan yang mampu membuka sesuatu yang terkunci, penutup dari semua yang terdahulu, penolong kebenaran dengan jalan yang benar, dan petunjuk kepada jalan-Mu yang lurus.

Skripsi dengan judul "PENGEMBANGAN DAN IMPLEMENTASI *BACKEND & REST-API DASHBOARD* IOT PADA *INTELLIGENCE CONTROL SYSTEM (ICS) SMART GREENHOUSE* MENGGUNAKAN PROTOKOL MQTT" ini merupakan salah satu syarat untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro, Jurusan Teknik Elektro, Fakultas Teknik, Universitas Lampung.

Pada kesempatan ini, penulis mengucapkan terimakasih kepada:

1. Allah SWT yang telah memberikan rahmat, karunia, rezeki, kemudahan dan ilmu-Nya kepada penulis sehingga penulis mampu menyelesaikan skripsi ini.
2. Kedua Orangtua tercinta Bapak Asmaranur dan Ibu Aniyar, terimakasih atas segala doa, cinta, kasih sayang, bimbingan, arahan dan perjuangan dalam memberikan dukungan baik moril dan materil yang sangat berharga kepada penulis.
3. Saudara-saudara penulis Adzkia Celina Asmara dan Ashila Permata Asmara yang telah memberikan doa dan semangat kepada penulis untuk menyelesaikan pendidikan dan membuat bangga kedua orangtua.
4. Ibu Prof. Dr. Ir. Lusmeilia Afriani, D.E.A., I.P.M., selaku Rektor Universitas Lampung.
5. Bapak Dr. Eng. Ir. Helmy Fitriawan, S.T., M.Sc. selaku Dekan Fakultas Teknik Universitas Lampung.

6. Ibu Herlinawati, S.T., M.T. selaku Kepala Jurusan Teknik Elektro Universitas Lampung.
7. Ibu Yetti Yuniati, S.T., M.T. selaku dosen pembimbing utama yang telah memberikan bimbingan, arahan, dan nilai-nilai kehidupan kepada penulis dengan baik dan ramah. Dan membantu penulis mempersiapkan diri menjadi seorang sarjana teknik.
8. Bapak Dr. Ing. Ardian Ulvan, S.T., M.Sc.. selaku dosen pembimbing akademik (PA) dan selaku dosen pembimbing pendamping yang telah memberikan nasihat, arahan, dan bimbingan rutin, motivasi, dan pandangan kehidupan kepada penulis di setiap kesempatan dengan baik dan ramah dalam mempersiapkan diri menjadi seorang Sarjana Teknik.
9. Ibu Dr. Ing. Melvi, S.T., M.T selaku dosen penguji yang telah memberikan kritik dan saran yang membangun kepada penulis dalam mengerjakan skripsi ini.
10. Segenap Dosen di Jurusan Teknik Elektro yang telah memberikan ilmu yang bermanfaat, wawasan, dan pengalaman bagi penulis.
11. Mbak Nurul dan Segenap Staff di Jurusan Teknik Elektro dan Fakultas Teknik yang telah sangat membantu penulis baik dalam hal administrasi dan hal-hal lainnya.
12. Keluarga besar HELIOS 2020 dan teman teman PSTE 2020 terima kasih telah memberikan banyak bantuan dan motivasi serta sudah menjadi keluarga selama awal kuliah sampai saat ini.
13. Kepada member KEQING WNGY. Gusti, Sandro, Dian, Irham, Sidik dan Fadhil. Terima kasih sudah menjadi teman berbincang serta berdiskusi berbagi kesenangan dan kegembiraan serta supportnya dari awal perkuliahan sampai masa masa akhir perkuliahan penulis juga seterusnya.
14. Teman Teman Pengurus HIMATRO UNILA yang selalu memberikan dukungan kepada penulis dalam penyelesaian skripsi.
15. Kak M. Nur Hasanuddin yang telah memberikan masukan dan arahan serta membimbing penulis dari awal hingga penulis dapat menyelesaikan skripsi ini.

16. Seluruh Tim Project PPK ORMAWA GISTING yang telah memberikan bantuan serta dorongan penulis dalam penyelesaian project yang dibuat.
17. Seluruh Tim Capstone Project AGRI-ICS yang telah memberikan bantuan serta dorongan penulis dalam penyelesaian project yang dibuat.
18. Semua pihak yang terlibat dalam proses perkuliahan dan penulisan skripsi ini penulis ucapkan terimkasih atas bantuan dan kerjasamanya.
19. Seluruh teman-teman yang terlibat langsung maupun tidak langsung dalam penyelesaian skripsi.
20. Terimakasih kepada diri penulis sendiri yang sudah berusaha dan yakin dalam mengerjakan apa yang telah dimulai, dan tidak menyerah terhadap impian dan cita-cita yang diimpikan. Terimakasih sudah berjuang dan berhasil menyelesaikan salah satu tahap dalam hidup, dan semangat dalam menghadapi tantangan-tantangan yang akan dihadapi kedepannya.

Penulis menyadari bahwa masih banyak kekurangan dalam penulisan skripsi ini. Penulis terbuka terhadap kritik dan saran yang membangun dari semua pihak demi kemajuan bersama. Penulis berharap skripsi ini dapat bermanfaat bagi penulis dan masyarakat.

Bandar Lampung, 7 September 2024

M. Affan Siddiqie Asmara

DAFTAR ISI

Halaman

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR LAMPIRAN

Halaman

BAB I
PENDAHULUAN

1.1. Latar Belakang

Indonesia, sebagai negara agraris, mengandalkan pertanian sebagai tulang punggung kehidupan, terutama di kalangan masyarakat kecil. Saat ini, kegiatan pertanian merajalela di berbagai komunitas di Indonesia, terutama di lingkungan masyarakat kecil. Namun, masyarakat kecil yang berada di daerah terpencil masih mengalami kendala akibat minimnya pemanfaatan dan pengembangan teknologi yang dapat mendukung pengelolaan lahan pertanian dan hasilnya. Sebagian besar petani masih sangat tergantung pada kondisi cuaca alam, sehingga hasil pertanian tidak selalu memuaskan ketika cuaca tidak sesuai harapan [1]. Aktivitas manusia untuk mempertahankan hidupnya juga berdampak signifikan pada peningkatan kebutuhan lahan. Akibatnya, lahan pertanian menjadi unsur krusial dalam mendukung kehidupan manusia. Pemanfaatan lahan ini semakin meningkat karena banyak orang menggunakan lahan untuk tempat tinggal, usaha, pembangunan akses umum, dan fasilitas lainnya, yang pada akhirnya mengakibatkan keterbatasan luas lahan. Penggunaan lahan yang tidak terkendali ini dapat mengakibatkan gangguan pada keseimbangan ekosistem [2].

Agar pemantauan data pada *smart greenhouse* dapat dilakukan dengan baik maka diperlukan sebuah *dashboard* untuk me*monitoring* data-data yang telah diterima dari sensor yang ada pada *smart greenhouse*. *Dashboard* adalah alat yang menyediakan antarmuka *visual*, yang menggabungkan dan menyajikan informasi penting untuk mencapai tujuan tertentu secara sekilas. Tampilan visual *dashboard* yang mampu mengkomunikasikan informasi dengan jelas, cepat, dan memberikan persepsi benar-benar menjadi kunci keberhasilan *dashboard*. Konsep visualisasi data dan informasi akan digunakan saat merancang antarmuka *dashboard*. Visualisasi data dan informasi terkait hal-hal mengenai persepsi visual dan media

penyajian data, penyampaian komponen *dashboard* harus mengutamakan efektifitas penyampaian informasi untuk memudahkan pengguna melihat, memantau dan membantu dalam mengambil keputusan yang tepat. dalam waktu nyata [3].

Dalam pengembangan *dashboard*, keberadaan *backend* sangat penting karena berperan sebagai pondasi yang mendukung pengolahan dan penyimpanan data secara efisien. *Backend* merupakan bagian dari sistem yang bertanggung jawab untuk mengelola basis data, menyimpan informasi dari sensor, dan mengolah data mentah menjadi informasi yang dapat ditampilkan dengan jelas pada *dashboard*.

Implementasi *backend* pada *dashboard* ini mengadopsi protokol MQTT (*Message Queuing Telemetry Transport*) sebagai salah satu komponen utama dalam sistem *backend*. Protokol MQTT dipilih karena kemampuannya untuk menyampaikan data secara ringan dan efisien melalui jaringan, yang sangat relevan untuk aplikasi *Internet of Things* (IoT) seperti *Smart Greenhouse*. MQTT memungkinkan perangkat di dalam *greenhouse* untuk mengirimkan data ke *backend* secara real-time, memastikan bahwa informasi yang disajikan pada *dashboard* selalu terkini. *Virtual Private Server* (VPS) digunakan sebagai lingkungan *hosting* untuk *backend*, dan *Node.js*, sebagai *platform* pengembangan *server-side* yang berbasis *JavaScript*, menjadi pilihan dalam mengimplementasikan *backend* pada penelitian ini.

1.2. Rumusan Masalah

Rumusan masalah dari penelitian ini adalah :

1. Bagaimana data yang dikirimkan oleh sensor-sensor yang digunakan pada ICS dapat diterima menggunakan protokol MQTT (*Message Queuing Telemetry Transport*)
2. Bagaimana menjalankan sistem *backend* pada *Virtual Private Server* (VPS)
3. Bagaimana data yang telah diterima dari sensor dapat disimpan dan diolah kedalam *database postgreeSQ* menggunakan *Node.js.*
4. Bagaimana cara data yang telah disimpan dapat diambil dan diolah menjadi sebuah API sebagai bahan penelitian lain.

1.3. Batasan Masalah

Adapun batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Penelitian berfokus pada mengimplementasikan protokol komunikasi *Message Queuing Telemetry Transport (MQTT)* pada *Virtual Private Server (VPS)* yang terhubung dengan ICS *Smart Greenhouse* dan mengirimkan data ke *database PostgreeSQL* menggunakan *Node.js*, dan perancangan API yang akan digunakan oleh *frontend* dan *mobile apps.*
2. Penelitian ini tidak mencakup proses pembuatan *dashboard* dan alat sensor *Intelligence Controlling System* pada *Smart Greenhouse.*

1.4. Tujuan Penelitian

Adapun tujuan penelitian adalah sebagai berikut:

1. Mengimplementasikan sebuah protokol komunikasi *Message Queue Telemetry Transport (MQTT)* dalam proses pengiriman dan menerima data sensor.
2. Menyimpan data yang diterima oleh sensor pada ICS *Smart Greenhouse* kedalam *database PostgreeSQL* menggunakan *Node.js*.
3. Membangun API yang dapat digunakan untuk mengambil data yang tersimpan di dalam *database*.

1.5. Manfaat Penelitian

Adapun manfaat dari penelitian adalah :

1. Optimalisasi kegiatan pertanian. Petani dan peneliti dapat memantau kondisi tanaman secara *real-time*, melalui data sensor yang dikirimkan oleh *smart* ICS
2. Memudahkan para peneliti terkait dan petani untuk mengolah data yang diperlukan.
3. Menyimpan data yang dikirimkan oleh *Intelligence Controlling System* kedalam sebuah *database* yang dapat diambil dan digunakan pada penelitian lebih lanjut.

1.6. Sistematika Penulisan

Sistematika penulisan dari penelitian ini adalah:

BAB I : PENDAHULUAN

Bab ini menjelaskan tentang latar belakang, tujuan penelitian, rumusan masalah, batasan masalah, manfaat penelitian, dan sistematika penulisan pada penelitian ini.

BAB II : TINJAUAN PUSTAKA

Membahas tentang penelitian-penelitian sebelumnya pada tinjauan pustaka, dan dasar-dasar teori dari penelitian pengembangan dan implementasi *backend dashboard* IoT pada *intelligence controlling system* (ICS) *smart greenhouse* menggunakan protokol MQTT

BAB III : METODOLOGI PENELITIAN

Menjelaskan waktu, tempat, alat dan bahan, dan metode penelitian beserta tahapannya mengenai pengembangan dan implementasi *backend dashboard* IoT pada *intelligence control system* (ICS) *smart greenhouse* menggunakan protokol MQTT. Penelitian dilakukan menggunakan metode SDLC dan pengujian pefroma API.

BAB IV : HASIL DAN PEMBAHASAN

Menjelaskan bagaimana proses pelaksanaan penelitian dan juga hasil yang didapatkan pada penelitian.

BAB V : KESIMPULAN DAN SARAN

Memaparkan kesimpulan apa saja yang didapat dari proses pelaksanaan penelitian dan memberikan saran untuk dilakukan pada penelitian selanjutnya.

DAFTAR PUSTAKA

Bab ini berisikan referensi dari penulisan dan pelaksanaan *penelitian*.

BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Peneliti mengambil beberapa penelitian terkait sebagai referensi ataupun acuan untuk penelitian yang dilakukan.

Penelitian oleh Eric Anthony dan Tony (2020) di Fakultas Teknologi Informasi Universitas Tarumanagara, mengembangkan aplikasi manajemen data publikasi dan penelitian menggunakan teknologi web seperti *HTML, CSS,* dan *JavaScript* untuk *frontend*, serta *Node.js* dan *PostgreSQL* untuk *backend*. Aplikasi ini dirancang untuk meningkatkan efisiensi pengelolaan data akademik dengan fitur seperti pencarian yang mudah, grafik statistik interaktif, dan integrasi dengan sistem identifikasi digital [4].

Penelitian terkait berikutnya dilakukan oleh Narges A-hussein dan Ayman D Salman pada tahun 2020, mengenai IoT *Monitoring System Based on* MQTT *Publisher/Subscriber Protocol*. Peneliti melakukan percobaan pengiriman data dari sensor DHT11 menggunakan NodeMCU ESP32 dengan metode *publish* dan *subscribe protocol* MQTT, data yang dipublish pada NodeMCU menggunakan *topic* kemudian *disubscribe* pada *Node-red* yang telah di *install* pada *Raspi* 3 [5].

Penelitian dilakukan oleh Dedy Hermanto, Axel Natanael Salim dan Ivan Putra Pratama pada tahun 2021, berjudul "Sistem Monitoring Suhu dan Kelembapan Udara Menggunakan Protokol MQTT Berbasis Wemos D1 Mini". Penelitian ini berfokus pada visualisasi data yang diterima dari dht11 melalui ESP8266 Menggunakan MQTT. Penelitian ini menggunakan *firebase* sebagai *database* untuk menyimpan dan mengolah data yang diterima [6].

Penelitian dilakukan oleh Rizqi Kartika Safitri dan Hanson Prihantoro Putro pada tahun 2022 mengenai implementasi *Rest-API* untuk komunikasi antara *ReactJS* dan *NodeJS*, penelitian ini menjelaskan bagaimana *Rest-API* digunakan sebagai

jembatan komunikasi dalam pengembangan modul manajemen user yang lebih baik saat menggunakan *ReactJS* dan *NodeJS*. Perancangan *Rest-Api* menghasilkan *endpoint* yang akan menjembatani komunikasi dengan *rest client* pada bagian *Front End* [7].

Dian Resha Agustina. dalam jurnal yang diterbitkan pada September 2023, "Implementasi *Service Data* untuk Pemantauan *Lighting* pada *Smart Agriculture*", mengungkapkan inovasi dalam penggunaan data terstruktur dan teknologi *Internet of Things (IoT)* untuk pemantauan pencahayaan dalam pertanian cerdas. Penelitian ini menggabungkan teknologi *Service Worker* untuk memfasilitasi aplikasi web menggunakan *Node.JS*, *MongoDb* dan *broker RabbitMQ* [8].

Tabel 2.1 State of the art

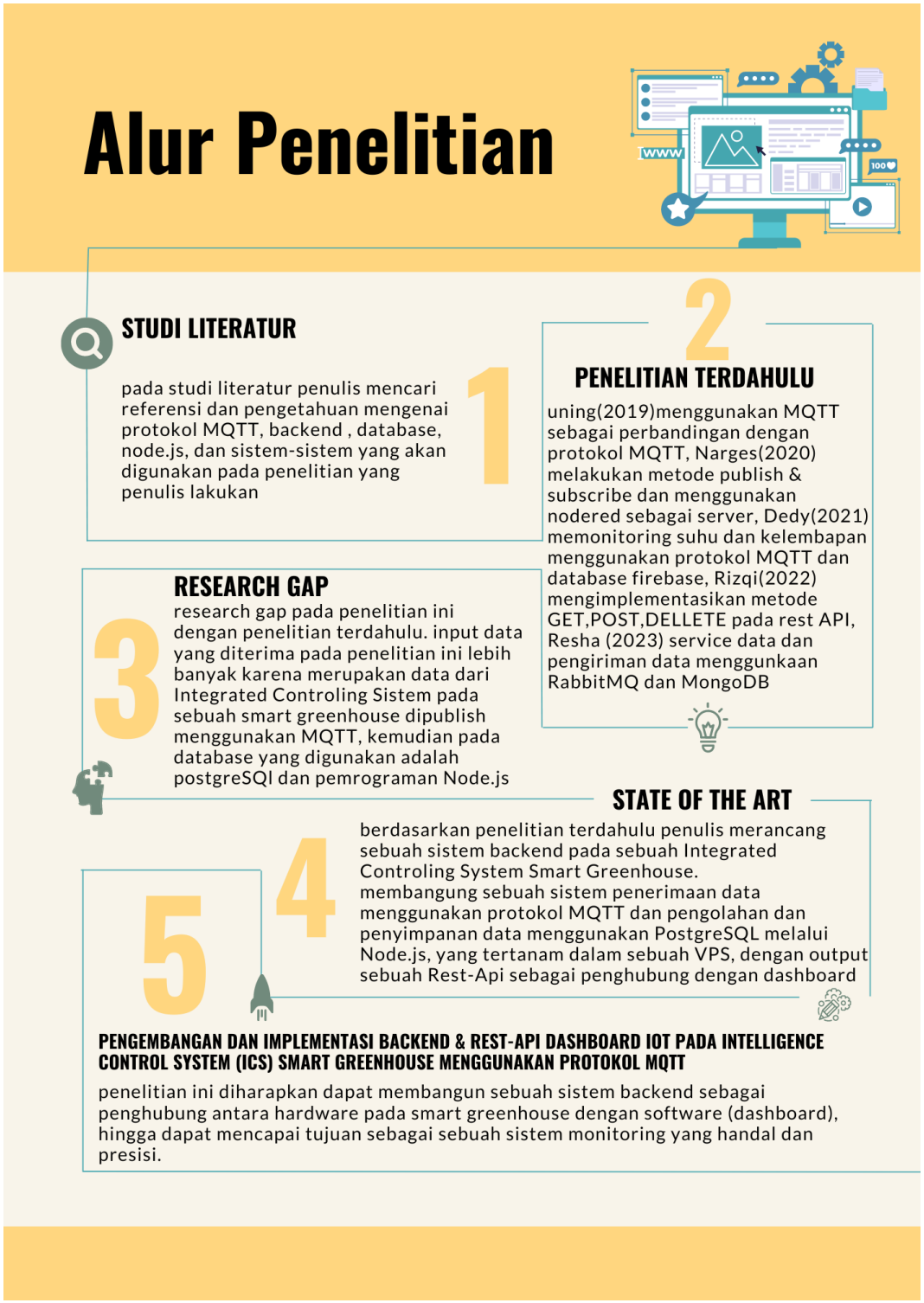
No	Judul Jurnal	Penulis dan Tahun	Perbedaan dengan Penulis
1	"Perancangan Aplikasi Manajemen Data Publikasi dan Penelitian"	Eric Anthony & Tony,2020	Penelitian dilakukan dengan merancang sebuah aplikasi manajemen data untuk publikasi dan penelitian pada Universitas Tarumanagara, dimana pada penelitian yang dilakukan membuat *front-end* dan juga *backend* menggunakan teknologi *HTML, CSS, Javascript* pada *front-end* dan *node.js, express.js* dan *PostgreSQL* pada *backend.* Aspek yang dirancang juga terkait dengan aksesibilitas kepada akademik univesitas. Sedangkan pada penelitian yang penulis lakukan membahas *backend* sistem dimana implementasi dilakukan pada komunikasi dengan IoT menggunakan *MQTT*

No	Judul Jurnal	Penulis dan Tahun	Perbedaan dengan Penulis
			protocol untuk perancangan *dashboard monitoring smart greenhouse*.
2	*"IoT Monitoring System Based on MQTT Publisher/Subscriber Protocol"*	Narges A-hussein & Ayman D Salman, 2020	Pada penelitian berfokus pada tahapan yang dilakukan saat mengguankan *protocol* MQTT, menjelasakan setiap tahapan-tahapan yang dilakukan jika menggunakan *protocol* MQTT yaitu *publish* dan *subscribe*, kemudian data yang diterima *disubscribe* menggunakan *node-red* sebagai *web server*. Sedangkan peneliti melakukan *subscribe* pada *topic* menggunakan *programming* menggunakan *Node.js* dan menggunakan VPS *(Virtual Private Server)* sebagai *web server*.
3	"Sistem *Monitoring* Suhu Dan Kelembapan Udara Menggunakan Protokol Mqtt Berbasis *Wemos D1 Mini"*	Dedy Hermanto, Axel Natanael Salim & Ivan Putra Pratama, 2021	Pada penelitian ini dilakukan *monitoring* suhu dan kelembapan udara yang dikirimkan oleh sensor DHT11 menggunakan *protocol* MQTT kemudian data tersebut disimpan kedalam *database*. Perbedaan penelitian ini dengan yang peneliti lakukan adalah pada parameter-parameter data *sensor* yang diterima dimana dan pada *database* yang digunakan untuk menyimpan data dimana pada penelitian ini

No	Judul Jurnal	Penulis dan Tahun	Perbedaan dengan Penulis
			menggunakan *database firebase* sedankan penelitian yang peneliti lakukan menggunakan *PostgreSQL*.
4	"Implementasi *REST API* untuk Komunikasi Antara *ReactJS* dan *NodeJS* (Studi Kasus : Modul Manajemen *User* Solusi247)"	Rizqi Kartika Safitri dan Hanson Prihantoro Putro, 2022	Penelitian ini melakukan implementasi terhadap API yang dirancang dengan arsitektur *REST* digunakan tiga metode yaitu *GET, POST, DELETE*. Sedangkan pada penelitian yang peneliti lakukan metode *REST* yang digunakan adalah *GET*.
5	"Implementasi *Service* Data untuk Pemantauan *Lighting* pada *Smart Agriculture*"	Dian Resha Agustina, 2023	Penelitian ini melakukan implemantasi *service data* untuk pemantuan pada *lightning* yang ada pada *smart agriculture*, pada penelitian ini digunakan metode pengiriman data menggunakan *RabbitMQ* dan *database MongoDb*, sedangkan pada penelitian yang penulis lakukan menggunakan *protocol MQTT* pada pengiriman datanya dan *PostgreSQl* sebagai *database*

Berdasarkan Tabel 2.1 penelitian pertama terkait penggunan teknologi seperti *Node.js, express.js* dan *database postgreSQL*. Penelitian kedua berfokus pada implementasi MQTT dalam sistem monitoring IoT, dengan penekanan pada tahapan *publish* dan *subscribe*, serta penggunaan *Node.js* dan VPS sebagai *web server*. Dalam konteks *monitoring* suhu dan kelembapan, penelitian ketiga menggunakan sensor DHT11 dan protokol MQTT, dengan penyimpanan data pada *database Firebase*, berbeda dengan penelitian keempat yang menggunakan

PostgreSQL. Terkait implementasi *REST API*, penelitian menyoroti penggunaan metode *GET, POST,* dan *DELETE* dalam komunikasi antara *ReactJS* dan *NodeJS*. Dalam konteks *smart agriculture*, penelitian kelima mengimplementasikan *service* data untuk pemantauan *lighting* menggunakan *RabbitMQ* dan *MongoDB*.



Gambar 2.1 *Roadmap* penelitian

2.2. *Smart Greenhouse*

Smart Greenhouse merupakan evolusi dari rumah kaca tradisional, di mana teknologi sensor dan otomatisasi digunakan untuk meningkatkan efisiensi dan pengendalian lingkungan pertanian. Dengan memanfaatkan sensor suhu, kelembaban, intensitas cahaya, dan CO2, serta sistem otomatisasi seperti pengairan dan pengaturan suhu [9]. *Smart Greenhouse* menawarkan pendekatan terintegrasi untuk meningkatkan produktivitas dan keberlanjutan dalam pertanian. Sistem ini memungkinkan pengguna untuk memantau dan mengontrol kondisi pertumbuhan tanaman secara *real-time*, membuat penyesuaian berdasarkan data, dan bahkan melakukan tindakan otomatis untuk menjaga kondisi yang optimal bagi tanaman.

2.3. *Intelligence Controlling System (ICS)*

Intelligence Controlling System (ICS) adalah sistem terpadu yang dirancang untuk mengoptimalkan dan mengontrol berbagai aspek dalam pertanian, termasuk pemantauan lingkungan, irigasi, pemupukan, dan manajemen sumber daya secara menyeluruh. ICS menggunakan teknologi sensor, dan konektivitas berbasis *Internet of Things* (IoT) untuk memberikan solusi terpadu yang memungkinkan petani untuk mengambil keputusan berdasarkan data secara akurat dan efisien. Salah satu elemen kunci dari ICS adalah integrasi data dari berbagai sensor yang terdistribusi di seluruh lahan pertanian. Sensor ini dapat mencakup sensor suhu, kelembaban tanah, tingkat nutrisi, dan lainnya. Data yang dikumpulkan oleh sensor-sensor ini kemudian diolah oleh sistem untuk memberikan pemahaman yang lebih mendalam tentang kondisi pertanian.

2.4. Internet Of Things (IoT)

Internet of Things (IoT) merupakan sebuah konsep yang sangat populer di masa ini dimana sebuah perangkat ataupun objek dapat ditanamkan sebuah teknologi seperti sensor yang dapat berkomunikasi, mengendalikan, menghubungkan dan bertukar data dengan perangkat lain dengan memanfaatkan internet.

Pada dasarnya IoT adalah sebuah ekosistem yang dilakukan secara berkala dalam pengambilan data dari sebuah perangkat ataupun sensor yang kemudian data tersebut dikirimkan menggunakan jaringan internet sebagai jalur komunikasi antara perangkat dengan *server* IoT, dimana *server* dapat berperan sebagai penyimpanan data. Manfaat yang didapat dalam penggunaan *Internet of Things* (IoT) ialah membuat pekerjaan yang dilakukan dapat menjadi lebih cepat, mudah dan efisien sehingga hal ini dapat meningkatkan kinerja penggunanya [10].

Pada penelitian ini, pengemembangan sistem yang dilakukan dalam penelitian ini menerapkan konsep IoT, dimana terdapat sebuah sensor pemantauan yang terhubung dengan internet, dan sensor tersebut dapat diakses datanya secara jarak jauh menggunakan jaringan internet.

2.5. Message Queue Telemetry Transport (MQTT)

Message Queue Telemetry Transport (MQTT) adalah sebuah protokol standar baru yang banyak digunakan dalam pengembangan sebuah sistem yang berbasis *Internet of Things (IoT).* Sistem ini sangat cocok dalam penggunaannya pada perangkat IoT dikarenakan protokol ini sangat ringan, mudah digunakan, dan hemat energi. *MQTT* dalam pengembangannya saat ini sudah banyak digunakan dalam berbagai industry seperti otomotif, manufaktur, telekomunikasi, minyak dan gas.

MQTT saat ini menjadi pilihan sebagai jalur komunikasi IoT dikarenakan beberapa kelebihan seperti :

1. Ringan dan Efisien.
   MQTT-clients yang sangat ringan, hanya membutuhkan sedikit sumberdaya dalam penggunaan nya sehingga dapat digunakan dalam *mikrokontroller* dengan memori berukuran kecil. Dalam pengiriman datanya, *MQTT Message Header* yang berfungsi sebagai identitas alamat komunikasi juga sangat ringan sehingga penggunaan *network bandwidth* juga sangat optimal.
2. Pengiriman pesan yang reliabel.
   Reliabilitas dalam pengiriman pesan sangat penting dalam berbagai perangkat IoT. *MQTT* memiliki 3 level *Quality of Service (QoS)* yang dapat disesuaikan.

3. Komunikasi dua arah.

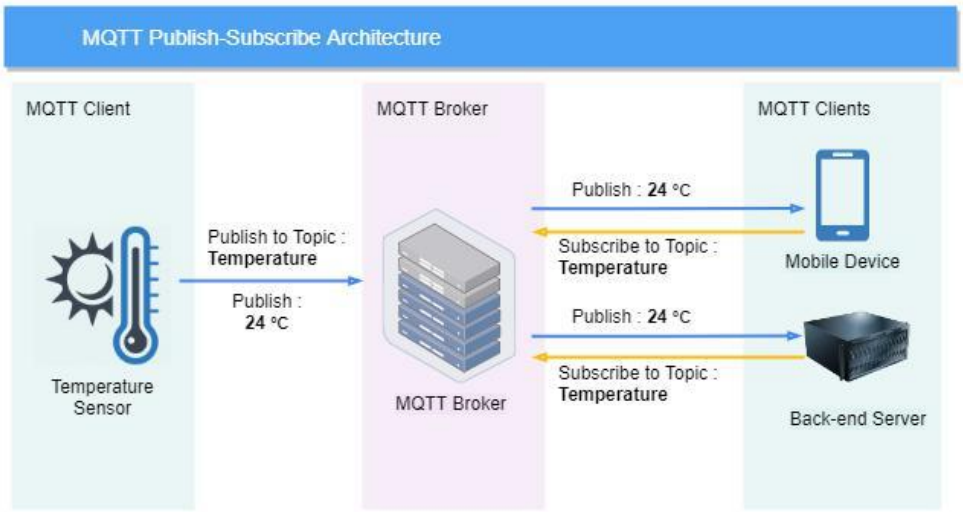
   MQTT dapat melakukan pengiriman pesan antara *device* ke *cloud* dan juga *cloud* ke *device*. Hal ini memudahkan membagikan pesan dalam group dengan *broadcasting*.
4. Dapat berjalan dalam jaringan yang kurang terkendali.

   Banyak perangkat IoT terhubung melalui jaringan seluler yang kurang terkendali. *MQTT* dapat dijalankan secara persisten sehingga mengurangi waktu untuk menghubungkan kembali antara klien dengan *broker*.
5. Skala besar.

   MQTT dapat terhubung dengan jutaan perangkat IoT.
6. Keamanan.

   MQTT memudahkan untuk mengenkripsi pesan menggunakan TLS dan mengautentikasi klien menggunakan *protocol* autentikasi modern.

Cara kerja dari *MQTT* adalah dengan menggunakan metode *publish / subsribe.* Metode ini bekerja dengan cara satu perangkat mengirimkan sebuah pesan (*publish*), kemudian perangkat lain yang bertindak sebagai penerima (*subscribe*) dapat menerima pesan yang telah dikirimkan tadi sesuai dengan *topic* yang sesuai. Seluruh pengiriman dan penerimaan pesan ini diatur oleh *MQTT Broker* [11]. Cara kerja dari MQTT dapat dilihat pada Gambar 2.2.



Gambar 2.2. Cara Kerja MQTT [21]

Pada sistem yang akan dibangun dalam penelitian ini, *MQTT* yang digunakan sebagai *broker* adalah *Mosquitto*. Program *MQTT* tersebut akan di*install* ke sebuah *Virtual Private Server (VPS)* dan dapat diakses dari luar jaringan *VPS* sebagai protokol komunikasi dan pengiriman data. Pemilihan menggunakan Broker *MQTT Mosquitto* adalah karena kelebihannya yang ringan, mudah untuk digunakan, gratis, *Open-source*, dan tersedia di berbagai *platform*.

2.6. Mosquitto MQTT

Mosquitto adalah sebuah perangkat lunak *open-source* yang berfungsi sebagai *message broker* untuk protokol *MQTT (Message Queuing Telemetry Transport)*. *Mosquitto* dikembangkan oleh *Eclipse Foundation* dan tersedia untuk digunakan secara bebas dengan lisensi EPL/EDL [12].

Mosquitto dirancang untuk menjadi sebuah *message broker* yang ringan dan sederhana, sehingga cocok digunakan pada berbagai jenis perangkat, dari komputer papan tunggal hingga *server*. *Mosquitto* juga menyediakan perpustakaan C untuk mengimplementasikan klien *MQTT*, serta klien *MQTT* pada baris perintah seperti *mosquitto_pub* dan *mosquitto_sub* yang sangat populer.

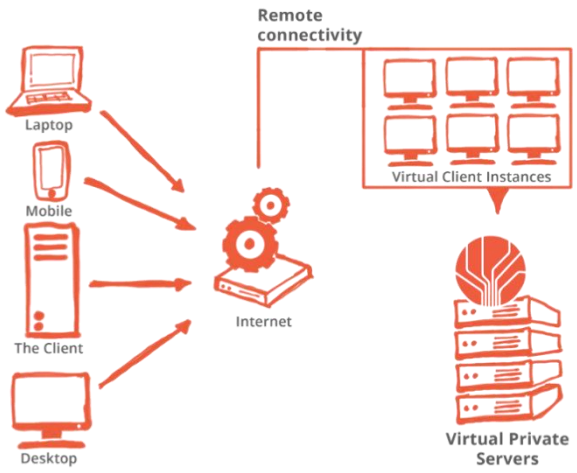
Gambar 2.3. MQTT *Broker Mosquitto* [22]

Mosquitto mendukung beberapa versi protokol *MQTT*, yaitu versi 5.0, 3.1.1, dan 3.1, sehingga dapat digunakan pada berbagai jenis aplikasi. Protokol *MQTT* yang digunakan oleh *Mosquitto* menggunakan model *publish*/*subscribe* yang sangat efisien untuk mengirim dan menerima pesan, dan dapat digunakan untuk berbagai jenis aplikasi, termasuk pada *Internet of Things (IoT)* dan perangkat berdaya rendah.

Beberapa fitur penting dari *Mosquitto* antara lain adanya dukungan untuk autentikasi dan enkripsi, adanya dukungan untuk *multiple broker*, serta kemampuan untuk mengatur *quality of service* (QoS) pada pengiriman pesan. *Mosquitto* juga dapat diintegrasikan dengan berbagai jenis perangkat dan *platform*, sehingga menjadi solusi yang fleksibel dan dapat disesuaikan dengan kebutuhan pengguna.

2.7. Virtual Private Server (VPS)

Virtual Private Server (VPS) merupakan sebuah teknologi virtualisasi dari sebuah *physical server*. Beberapa *Virtual Private Server* dapat dibuat dalam sebuah *physical server*. *VPS* dapat berjalan seperti sebuah *server* mandiri yang memiliki *full root access*, sistem operasi, konfigurasi, *user*, pemrosesan, *CPU*, dan *RAM* nya masing-masing sehingga sebuah *VPS* tidak akan mempengaruhi *VPS* lain dalam satu *physical server* [13].



Gambar 2.4. *Virtual Private Server* [23]

Pada penelitian digunakan sebuah *VPS (Virtual Private Server)* dikarenakan dalam membangun sebuah protokol komunikasi pengiriman data dibutuhkan sebuah layanan yang dapat beroperasi terus menerus. Apabila tidak menggunakan *VPS* melainkan menggunakan komputer/laptop secara lokal, terdapat kemungkinan bahwa perangkat tersebut tidak dapat dinyalakan secara terus menerus sehingga *protocol* komunikasi data juga dapat terganggu. Penggunaan *VPS* pada penelitian ini juga sangat penting dikarenakan seluruh proses pengiriman dan penyimpanan data akan dilayani oleh *VPS* secara terus menerus.

2.8. Backend Development

Pengembangan *Backend* pada *Dashboard* IoT *Smart Greenhouse* mencakup pembangunan infrastruktur yang mendukung fungsi dan integrasi semua komponen dalam sistem. *Dashboard* ini menjadi jantung operasional yang mengelola data dari sensor, memproses informasi, dan menyajikan hasilnya dalam bentuk yang dapat dimengerti oleh pengguna. Fungsi utama dari *Backend* pada *Dashboard* IoT *Smart Greenhouse* melibatkan:

1. Manajemen Data Sensor: *Backend* mengatur penerimaan dan penyimpanan data dari sensor-sensor yang terpasang di dalam *greenhouse*.

2. Integrasi dan Pemrosesan Data: Data yang dikumpulkan dari sensor diolah dan diintegrasikan oleh *Backend* untuk memberikan gambaran keseluruhan tentang kondisi lingkungan dalam *greenhouse*.
3. Interaksi dengan Database: *Backend* berkomunikasi dengan *database* untuk menyimpan dan mengelola data secara efisien. Ini memastikan keberlanjutan dan ketersediaan data untuk dianalisis atau ditampilkan pada *dashboard*.
4. API untuk Koneksi *Frontend*: *Backend* menyediakan API untuk menghubungkan *Dashboard* dengan *Frontend* (antarmuka pengguna).

2.9. *Node.js*

Node.js adalah sebuah *platform runtime JavaScript* yang dibangun di atas mesin *JavaScript* V8 milik Google yang berguna untuk proses *development* aplikasi secara cepat dan efisien serta mempermudah pembangunan aplikasi berbasis jaringan yang memiliki *scalability* (daya pengembangan) yang tinggi, Node Js menggunakan *event-driven non-blocking* I/O model yang membuat *Node.Js* ringan dan efisen, cocok untuk aplikasi *data-intensive realtime* yang berjalan pada *cross-platform* [14].

Gambar 2.5. *Node.js* [24]

Dengan menggunakan *Node.js*, pengembang dapat menulis kode *JavaScript* yang dapat dijalankan di sisi *server*, mengakses sistem file, dan melakukan tugas lainnya yang memerlukan akses ke sumber daya sistem. Beberapa manfaat yang ditawarkan oleh *Node.js* adalah kecepatan eksekusi, skalabilitas, kemampuan untuk mengatasi banyak permintaan (*concurrency*), dan kemudahan penggunaan.

2.10. *Express.js*

Express.js adalah kerangka kerja *web* yang dirancang untuk mempermudah pengembangan aplikasi *web* dengan menggunakan bahasa pemrograman *JavaScript* di sisi *server*. Dibangun di atas *Node.js, Express.js* menyediakan cara yang cepat dan minimalis untuk membuat *server* HTTP dan mengelola rute aplikasi. Salah satu kekuatan *Express.js* adalah pendekatannya yang ringkas, memberikan fleksibilitas kepada pengembang untuk membangun aplikasi dengan struktur yang mudah dimengerti [15].

Gambar 2.6. *Express.js* [25]

Express.js memudahkan penanganan permintaan HTTP dengan memungkinkan definisi rute-rute yang menentukan cara *server* menanggapi permintaan tertentu. Ini memungkinkan pengembang untuk membuat aplikasi *web* dengan mudah, termasuk pembuatan *endpoint API, pengaturan middleware*, dan manajemen sesi. Selain itu, *Express.js* memiliki sistem *template* yang dapat diintegrasikan dengan berbagai mesin *template*, memfasilitasi pembuatan tampilan dinamis.

Kerangka kerja ini juga mendukung pengembangan aplikasi *RESTful* dengan menyediakan alat untuk mengelola metode HTTP seperti *GET, POST, PUT*, dan *DELETE*. *Express.js* juga mendukung penggunaan *middleware*, yang dapat digunakan untuk menangani berbagai aspek seperti otentikasi, log, dan manipulasi permintaan sebelum mencapai penanganan utama.

2.11. *PostgreSQL*

PostgreSQL merupakan salah satu sistem basis data yang tergolong kedalam DBMS atau *Database Management System* yaitu suatu sistem perangkat lunak yang

kegunaanya didesain untuk membantu pekerjaan penggunanya seperti mengelola suatu basis data dan menjalankan suatu perintah operasi terhadap data yang diminta oleh banyak penggunanya. *PostgreSQL* digunakan sebagai manajemen basis data yang menampung dan mengolah data aplikasi dan dapat dijalankan pada sistem operasi seperti *Windows, Linux, Mac,* dan lain sebagainya [16].

Gambar 2.7. *PostgreeSQL* [26]

PostgreSQL dapat digunakan sebagai aplikasi manajemen *database* dengan aplikasi bawaannya *pgAdmin*. *PostgreSQL* bekerja dengan menangani proses pengolahan data dengan *sintaks query* SQL.

PostgreSQL dikenal sebagai sistem pengolahan *database open source* termaju di dunia yang umum digunakan dalam pengembangan perangkat lunak dengan kemampuannya yang mampu memproses banyak *sintaks query* dalam satu waktu.

PostgreSQL dipilih dalam penelitian ini sebagai *database* penyimpanan data dikarenakan beberapa kelebihan dari PostgreSQL yaitu *Open-source*, skalabilitas nya yang baik, mudah dikonfigurasi, dapat menangani volume data yang besar, dan lebih cepat dibandingkan beberapa *Database Management System* (DBMS) yang lain seperti *MySQL* atau *MongoDB*.

2.12. *Application Programming Interface (API)*

Application Programming Interface (API) adalah sebuah antarmuka yang memungkinkan aplikasi untuk berinteraksi dengan aplikasi atau layanan lainnya

secara programatik. API menyediakan kumpulan instruksi dan protokol yang dapat digunakan oleh pengembang untuk membangun aplikasi yang dapat berkomunikasi dengan sistem atau layanan yang telah ada [17].

API memungkinkan pengembang untuk mengakses fungsi dan layanan yang disediakan oleh sistem atau layanan lainnya tanpa harus mengetahui seluruh detail implementasi di belakangnya. Dengan menggunakan API, pengembang dapat mempercepat proses pengembangan, mengurangi biaya, dan meningkatkan efisiensi dalam pengembangan aplikasi karena tidak perlu membangun seluruh fitur dari awal.

Contoh penggunaan API adalah ketika sebuah aplikasi mengakses data dari layanan seperti *Facebook*, *Twitter*, atau *Google Maps*. API yang disediakan oleh layanan tersebut memungkinkan aplikasi untuk mengambil data dari layanan tersebut dan menggunakannya dalam aplikasi mereka sendiri.

2.13. REST-API

REST-API adalah salah satu jenis dari API yang didasarkan pada arsitektur *REST (Representational State Transfer)*. *REST* sendiri adalah sebuah arsitektur perangkat lunak yang terdiri dari aturan dan konvensi yang digunakan untuk membuat *web service*.

Dalam *REST API*, setiap sumber daya (*resource*) diidentifikasi dengan URI (*Uniform Resource Identifier*) dan diakses melalui metode HTTP seperti *GET*, *POST, PUT, DELETE*, dan sebagainya.

Selain itu, REST API juga menggunakan format data yang ringan dan terstruktur, seperti JSON (*JavaScript Object Notation*) atau XML (*eXtensible Markup Language*) untuk pertukaran data antara aplikasi dan *server*. Format data ini memudahkan pengolahan data oleh aplikasi klien (*client*) yang mengakses API.

Dalam penggunaannya, *REST-API* sering digunakan oleh pengembang aplikasi untuk mengintegrasikan berbagai aplikasi atau layanan secara terdistribusi dan

scalable. Selain itu, *REST-API* juga sering digunakan dalam pembuatan aplikasi *mobile*, *Internet of Things* (IoT), dan integrasi antar sistem yang berbeda *platform*.

2.14. GitHub

GitHub adalah *platform hosting* kode sumber berbasis *web* yang sangat populer di kalangan pengembang perangkat lunak. *Platform* ini menyediakan repositori *git* publik dan pribadi yang memungkinkan pengguna untuk menyimpan, mengelola, dan berkolaborasi dalam pengembangan perangkat lunak secara terdistribusi. *GitHub* memiliki fitur-fitur kolaborasi yang kuat, yang memungkinkan tim pengembang bekerja bersama dalam proyek perangkat lunak.

2.15. Visual Studio Code

Visual Studio Code adalah editor teks gratis dan *open-source* yang dikembangkan oleh *Microsoft. Visual Studio Code* memungkinkan pengguna untuk menulis kode pada berbagai bahasa pemrograman, termasuk HTML, CSS, *JavaScript*, PHP, *Python*, dan lain-lain. *Visual Studio Code* berjalan pada sistem operasi *Windows, Mac, dan Linux*. Editor teks ini dilengkapi dengan fitur-fitur seperti fitur *debugging*, *Git integration*, *autocomplete*, *snippet*, *syntax highlighting*, dan lain-lain .

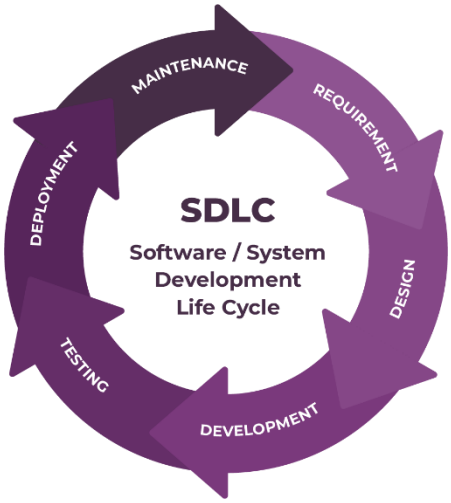
Visual Studio Code memiliki beberapa kelebihan yang membuatnya populer di kalangan pengembang perangkat lunak. Kelebihan-kelebihan tersebut antara lain adalah :

- Gratis dan *open-source*
- Ringan dan cepat
- Mudah digunakan
- Mendukung banyak bahasa pemrograman
- Memiliki banyak ekstensi dan plugin

2.16. *Software Development Life Cycling (SDLC)*

SDLC, singkatan dari *Life Cycle of Software Development*, atau SDLC, adalah metode yang digunakan dalam pengembangan perangkat lunak. Metode SDLC

mencakup model, metodologi, dan proses pembuatan dan pengubahan sistem. Metode ini juga digunakan dalam pengembangan sistem rekayasa perangkat lunak [18].



Gambar 2.8 SDLC [27]

Untuk membantu menghasilkan sistem yang berkualitas tinggi yang memenuhi tujuan atau permintaan, SDLC mencakup rencana lengkap pengembangan dan pemeliharaan perangkat lunak, yang mencakup alur atau tahapan kerja terstuktur. Metode SDLC memungkinkan pengembang mengukur dan meningkatkan proses pengembangan dengan lebih efektif karena memungkinkan analisis menyeluruh dari setiap tahapan proses.

Proses SDLC meliputi beberapa fase yang berbeda:

1. **Perencanaan (*Planning*):** Mengidentifikasi kebutuhan untuk perangkat lunak baru atau peningkatan perangkat lunak yang ada, termasuk analisis risiko dan penentuan sumber daya.
2. **Analisis Kebutuhan (*Requirements Analysis*):** Mendefinisikan kebutuhan rinci dari pengguna akhir dan sistem agar produk akhir dapat memenuhi ekspektasi pengguna.
3. **Desain (*Design*):** Membuat arsitektur sistem yang mencakup desain *database*, desain antarmuka pengguna, dan desain sistem atau aplikasi.

4. **Pengembangan (*Implementation or Coding*):** Fase ini melibatkan penulisan kode sesuai dengan desain yang telah ditentukan sebelumnya.
5. **Pengujian (*Testing*):** Memeriksa dan men-*debug* kode untuk memastikan produk bebas dari kesalahan dan sesuai dengan persyaratan bisnis.
6. **Penerapan (*Deployment*):** Merilis perangkat lunak ke lingkungan produksi sehingga pengguna dapat mulai menggunakan produk.
7. **Pemeliharaan dan Dukungan (*Maintenance and Support*):** Melakukan perbaikan, peningkatan, dan penyesuaian pada perangkat lunak untuk meningkatkan kinerja atau mengadaptasi perubahan kebutuhan pengguna.

SDLC memastikan bahwa proses pengembangan perangkat lunak berjalan lancar dan efisien, dan membantu dalam menghasilkan produk perangkat lunak yang berkualitas yang memenuhi atau melebihi harapan pengguna. Model SDLC yang populer termasuk *Waterfall, Agile, Spiral,* dan *Iterative.*

2.17. Postman

Postman adalah sebuah alat yang digunakan oleh para pengembang perangkat lunak untuk berinteraksi dengan API (*Application Programming Interface*). API seperti pelayan di restoran, yang mengambil pesanan Anda (permintaan data atau fungsi) dan memberikan balasan dari dapur (sistem atau aplikasi). Dengan *Postman*, pengembang bisa menguji API dengan mengirimkan permintaan-nya dan melihat apa yang dikembalikan. Misalnya, jika Anda ingin melihat data pengguna dari sebuah layanan, Anda bisa "meminta" data tersebut melalui *Postman*, dan layanan akan "memberikan" data yang diminta .

Postman sendiri kemudian digunakan untuk mengirim permintaan ke API. “Permintaan dapat mengambil (*GET*), menambah (*POST*), menghapus (*DELETE*), atau memperbarui data (*PUT*)”. Permintaan dapat digunakan untuk mengirim parameter, detail *login*, informasi otorisasi, atau data tubuh lain yang diperlukan. Perangkat lunak *Postman* dapat diunduh ke komputer pengguna.

2.17. Performance Testing

Performance testing adalah proses evaluasi kinerja dari sebuah sistem atau aplikasi di bawah beban kerja tertentu. Tujuan dari *performance testing* adalah untuk memastikan bahwa sistem dapat menangani volume kerja yang diharapkan tanpa mengalami penurunan performa yang signifikan. Ada beberapa jenis pengujian kinerja, termasuk *load testing, stress testing, endurance testing*, dan *spike testing*. Pengujian ini digunakan untuk mengidentifikasi batas maksimum sistem, mengukur *respons time*, *throughput*, dan penggunaan sumber daya seperti CPU dan memori [19]. Pada penelitian ini dilakukan *testing performance* bertipe *fixed test* pada *software postman*, dimana testing dilakukan dengan mencatat nilai *respon time* dan *error rate* dari API yang telah dirancang.

2.17. Throughput Testing

Throughput testing adalah bagian dari *performance testing* yang mengukur jumlah data atau jumlah transaksi yang dapat diproses oleh sistem dalam periode waktu tertentu. Pengujian *throughput* penting untuk menentukan kapasitas sistem dan memastikan bahwa aplikasi dapat menangani jumlah permintaan yang tinggi tanpa menurunkan kualitas layanan [20]. Pengujian ini sering dilakukan dalam konteks jaringan, aplikasi web, dan sistem penyimpanan data. *Throughput testing* dilakukan untuk mendukung nilai dari *respon time* dan *error rate* yang diperoleh pada *performance testing* yang telah dilakukan.

BAB III
METODE PENELITIAN

3.1. Waktu dan Tempat Penelitian

Penelitian ini dilakukan pada :

Waktu : Oktober 2023 – Juli 2024

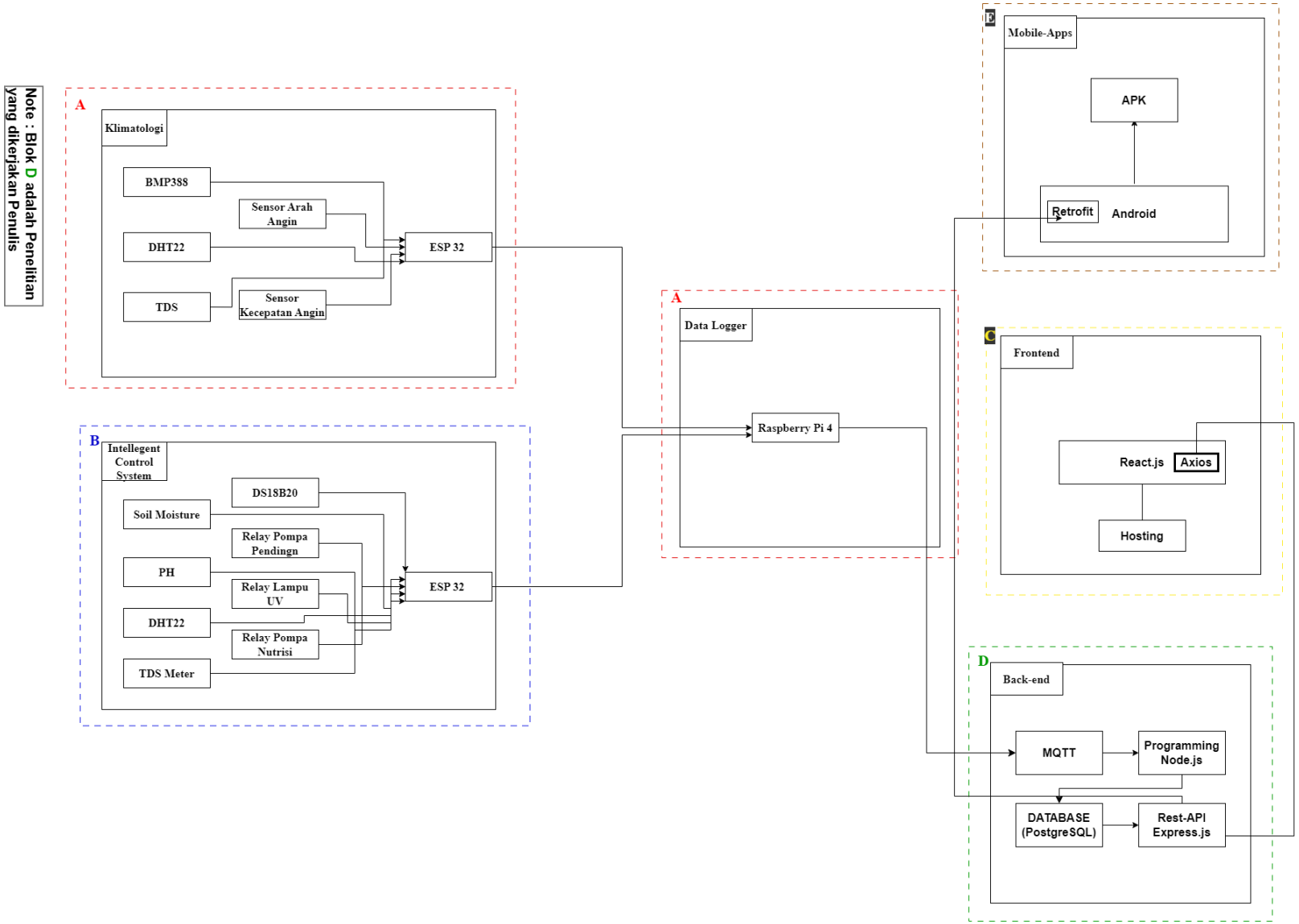
Tempat :

- Laboratorium Teknik Komputer, Teknik Elektro, Universitas Lampung
- Laboratorium Terpadu Fakultas Pertanian Universitas Lampung.

3.2. *Capstone Project*

Pengerjaan penelitian ini berkaitan dengan beberapa penelitian lain yang berkolaborasi untuk membangun sistem *Intelligence Controlling System* (ICS) pada *smart greenhouse*. Pada *project* yang akan dilakukan terdapat beberapa bagian yang akan dikerjakan secara terpisah. Pertama terdapat pengerjaan pada bagian *hardware* ICS *smart greenhouse* yang dikerjakan oleh dua orang dalam *team* ini, kemudian terdapat tiga orang yang meng*handle software* diantaranya adalah *frontend, mobile apps*, dan *backend* yang akan penulis kerjakan. Diagram keseluruhan *project* yang dikerjakan dapat dilihat pada Gambar 3.1.

Dalam pengerjaan proyek ini, penulis hanya akan berfokus pada bagian *Backend* yang ditandai dengan garis putus berwarna hijau dan bagan D, sementara bagian lain seperti perangkat ICS yang merupakan rangkaian *hardware* sensor pemantauan dan juga bagian *front-end* sebagai *dashboard monitoring* akan dikerjakan dalam penelitian lain. *Backend* dalam proyek ini berfungsi sebagai penghubung antara perangkat sensor dengan *front-end*, dimana didalam *server* itu sendiri akan menjalankan berbagai *service* yang dibutuhkan dalam proyek ini seperti *database*, *MQTT Broker*, dan *API*.



Gambar 3.1. Diagram keseluruhan pengembangan

3.3. Alat dan Bahan

3.3.1. Alat

Alat yang digunakan dalam pengerjaan penelitian dan pengembangan sistem yang dibuat adalah sebagai berikut :

Tabel 3.1. Alat berupa *Hardware* dan *Software* yang digunakan.

No	Perangkat	Spesifikasi	Kegunaan	Keterangan
1	*Laptop*	*Processor Intel Core* i5 10300H, RAM 16GB, SSD 512GB, Sistem Operasi *Windows*	Perangkat pembuatan dan pengujian sistem.	
2	*Virtual Private Server* (VPS)	Penyedia : Niagahoster.co.id, Lokasi: *United States*, 8 CPU *Core*, 32 GB RAM, Storage 400GB.	*Virtual private server* yang digunakan sebagai pusat layanan sistem yang akan dibuat.	Diakses melalui *Secure Shell (SSH)*
3	MQTTX	MQTTX Versi 1.9.6	*Software* yang digunakan untuk pengujian *broker* mqtt dalam mengirim dan menerima data	*Terinstall* di *Laptop*
4	*Visual Studio Code*	VS *Code* versi 1.84.2	*Software* yang digunakan dalam penulisan kode program untuk pengambilan dan penyimpanan data dari sensor	*Terinstall* di Laptop
3	*PostgreSQL*	*PostgreSQL database* versi 7.4	Program *database* yang digunakan dalam penyimpanan data dari sensor	*Terinstall* di VPS dan *laptop*

No	Perangkat	Spesifikasi	Kegunaan	Keterangan
4	*Mosquitto MQTT Broker*	*Mosquitto* versi 1.6.10	Program yang di *install* di dalam VPS sebagai *broker* MQTT	*Terinstall* di VPS dan laptop
5	*Node.js* & NPM	*Node.js* versi v16.19.0. NPM versi 8.19.3	Aplikasi yang digunakan untuk menjalankan kode program untuk penyimpanan data	*Terinstall* di VPS dan laptop
6	*Postman*	*Postman* V.10.19	Program yang digunakan untuk *checking Endpoint* API	*Terinstall* di Laptop

3.3.2. Bahan

Bahan yang digunakan dalam pengerjaan penelitian dan pengembangan sistem ini berupa data yang didapatkan dari sensor ICS adalah sebagai berikut :

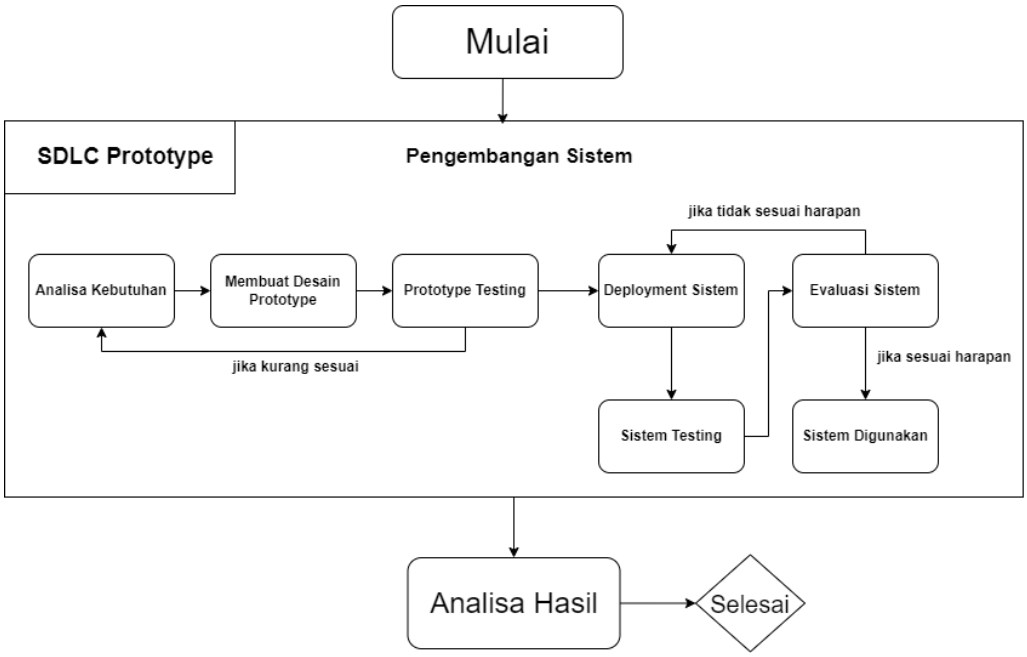
Tabel 3.2 Bahan berupa data sensor dari *esp*.

NO	Data yang diterima	Keterangan
1	*Timestamp*	Data yang menunjukan waktu
2	Ph	Data yang menunjukan tingkat keasaman atau kebasaan air
3	Tds	Data kualitas air
4	Suhu air	Data yang menunjukan suhu air
5	Arah mata angin	Data yang menunjukan arah mata angin pada greenhouse
6	Kecepatan angin	Data yang menunjukan kecepatan angin dalam *greenhouse*

NO	Data yang diterima	Keterangan
7	*Soilmoisture*	Data yang menunjukan kelembapan tanah pada tumbuhan
8	Suhu ruangan	Data yang menunjukan suhu *greenhouse*
9	Tekanan udara	Data yang menunjukan tekanan udara di dalam *greenhouse*
10	Pompa nutrisi	Data yang menunjukan apakah sistem pompa nutrisi menyala tau tidak
11	Pompa air	Data yang menunjukan apakah sistem pompa air menyala atau tidak
12	Kelembapan	Data yang menunjukan kelembapan didalam *greenhouse*

3.4 Tahapan Penelitian

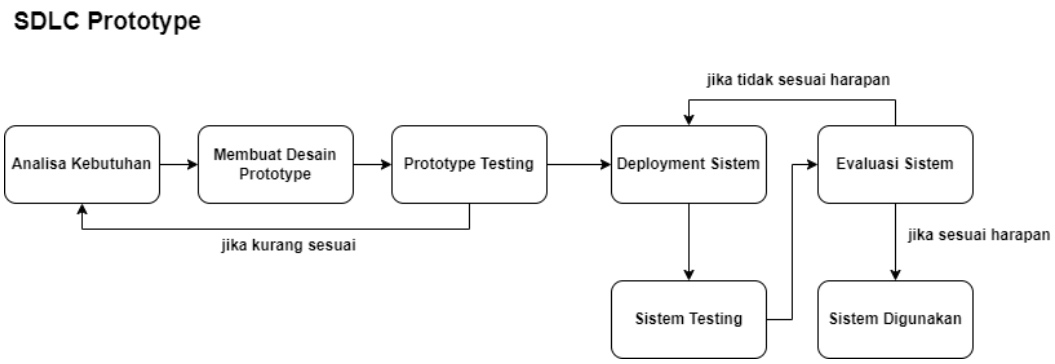
Pengerjaan dalam penelitian ini dilakukan secara bertahap mulai dari studi literatur lalu dilanjutkan dengan tahapan perancangan sistem menggunakan metode *Software Development Life Cycle (SDLC)* tipe *prototype*, kemudian setelah tahapan pengembangan sistem selelsai dilakukan analisa terhadap hasil yang didapatkan, tahapan-tahapan alir penelitian dapat dilihat pada Gambar 3.2.



Gambar 3.2 Diagram Blok Tahapan Penelitian

3.4.1 Pengembangan Sistem

Dalam penelitian ini, metode yang digunakan dalam pengembangan sistem akan mengikuti model pengembangan *Software Development Life Cycle (SDLC) Prototype*. Dimana pada akan dibagi menjadi beberapa tahapan pengembangan. Berikut ini adalah tahapan yang akan dilaksanakan dalam pengembangan menggunakan metode *SDLC*.



Gambar 3.3 Tahapan pengembangan sistem dengan Metode *SDLC Prototype*

Berdasarkan Gambar 3.3 Metode SDLC *prototype*, awalnya dilakukan analisis kebutuhan untuk memahami apa yang dibutuhkan oleh sistem. Berdasarkan pemahaman ini, dibuatlah desain awal atau *prototype*, yang kemudian diuji untuk memastikan kesesuaian dengan kebutuhan yang telah ditentukan. Proses pengujian ini membantu mengidentifikasi perbaikan yang diperlukan sebelum sistem dikembangkan lebih lanjut. Setelah tahap pengembangan, sistem di-*deploy*. Dan kemudian dilakukan proses *testing* terhadap sistem yang telah di-*deploy*, jika sistem bekerja sesuai dengan harapan maka akan sistem tersebut akan digunakan, sedangkan jika sistem tidak bekerja sesuai harapan atau ada terjadinya *error* maka dilakukan evaluasi pasca-*deployment* untuk memastikan kembali dan memperbaiki agar sistem bekerja seperti yang diharapkan. Setelah sistem memenuhi kriteria yang ditetapkan, tahap pengujian sistem lebih lanjut dilaksanakan untuk memastikan keandalan sebelum sistem secara resmi digunakan.

3.4.1.1 Analisa Kebutuhan

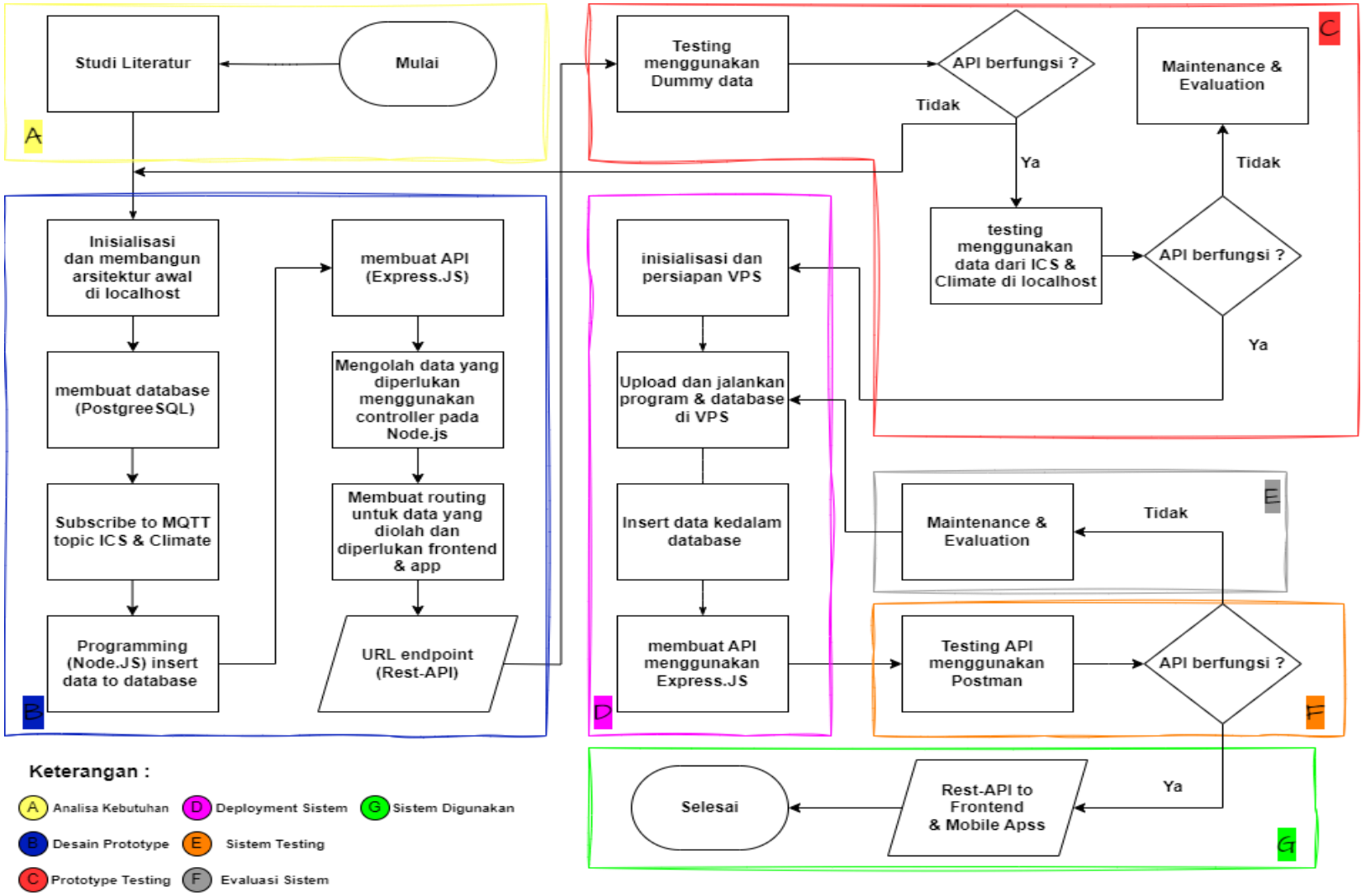
Penentuan kebutuhan didapatkan setelah mendapatkan beberapa data berupa literasi yang berkaitan dengan pengembangan dan implementasi *backend dashboard iot* pada *inteliggence control system (ICS) smart greenhouse* menggunakan protokol *MQTT* yang selanjutnya akan dicari informasi yang berhubungan dengan aktivitas yang akan dilakukan dalam pengembangan sistem.

Berdasarkan studi pustaka dan melihat kondisi ICS pada *smart greenhouse* ini telah berjalan, maka dapat ditentukan apa saja yang dibutuhkan dalam penelitian ini untuk mengatasi beberapa kelemahan yang ada pada sistem yang saat ini sedang berjalan.

Adapun setelah dilakukannya identifikasi terhadap tujuan pengembangan aplikasi dan kebutuhan, didapat beberapa kebutuhan yang dapat diselesaikan dengan penenilian ini antara lain :

1. *Virtual Private Server* yang dapat berfungsi sebagai *MQTT Broker* dan penyimpanan data.
2. *MQTT Broker* yang dapat digunakan sebagai protokol pengiriman data antara sensor pemantauan dengan *dashboard* dan *database*.
3. Data yang didapat dari sensor harus dapat disimpan dalam *database* VPS dan tidak boleh hilang dikarenakan data tersebut dapat digunakan sebagai data yang akan digunakan dalam *dashboard* dan sebagai bahan penelitian lebih lanjut.
4. Data yang tersimpan di *database* bisa dikirimkan ke *dashboard* atau *front-end* menggunakan *API.*

Berdasarkan beberapa kebutuhan tersebut, maka didapatkan beberapa *task* yang harus dilakukan. *Task* tersebut digambarkan pada Gambar 3.4:



Gambar 3.4 Tahapan pengembangan sistem

Proses pengembangan sistem dimulai dengan Studi Literatur untuk memahami dasar teori dan praktek terbaik yang relevan. Setelah memperoleh pengetahuan yang cukup, dilanjutkan dengan Inisialisasi dan Membangun Arsitektur Awal di *Localhost* yang meliputi pembuatan basis data menggunakan *PostgreSQL* dan pengaturan lingkungan pengembangan. Selanjutnya, dilakukan *Programming* dengan *Node.js* untuk *menginsert* data ke dalam *database*. Paralel dengan ini, sistem juga disiapkan untuk *Subscribe* to MQTT *topic* ICS & *Climate*, yang memungkinkan sistem untuk menerima data dari topik MQTT yang telah ditentukan.

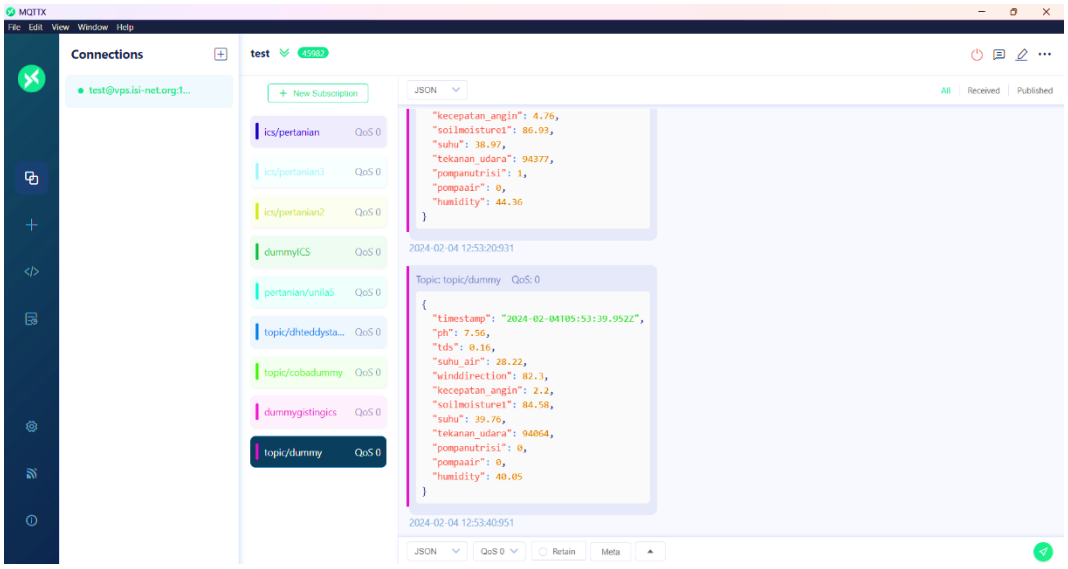
Setelah arsitektur lokal siap, proses berlanjut ke Inisialisasi dan Persiapan (*Virtual Private Server*) VPS di mana program dan *database* di-*upload* dan dijalankan di VPS. Data kemudian di*insert* ke dalam *database* di VPS. Sistem kemudian memasuki fase *Testing* menggunakan *Dummy* data untuk memastikan bahwa semua komponen berfungsi dengan benar. Jika API tidak berfungsi, maka perlu dilakukan pengecekan lagi terhadap tahapan yang dilakukan sebelumnya, jika berhasil selanjutnya dilakukan *testing* API menggunakan data dari ICS & *Climate* di *Localhost*. Setelah API terbukti berfungsi, dapat dilakukan pengujian lebih lanjut menggunakan *Postman* untuk memastikan bahwa API berfungsi sebagaimana mestinya.

Dengan API yang telah diverifikasi, tahap selanjutnya adalah mengembangkan URL *endpoint* menggunakan *Rest-API* yang akan digunakan oleh *frontend* dan aplikasi *mobile*. Setelah semua pengujian berhasil dan sistem stabil, proses pengembangan dianggap selesai, dan sistem siap digunakan dalam produksi. Seluruh proses ini dirancang untuk memastikan bahwa sistem yang dikembangkan tidak hanya memenuhi kebutuhan pengguna tetapi juga kuat, aman, dan skalabel.

3.4.1.2 Membuat Desain *Prototype*

Setelah mendapatkan hasil dari analisa kebutuhan tahap selanjutnya yang akan dilakukan adalah membuat desain *prototype*. Pada tahap ini membuat *prototype* awal pada arsitektur *localhost* merancang program yang telah disesuaikan dengan kebutuhan yang didapatkan dari tahapan analisa kebutuhan.

Berdasarkan Gambar 3.4 tahapan ini terletak pada bagian yang berada di kolom B atau kolom yang bergaris biru, dimana pada tahapan ini dilakukan inisialisasi dan membangun arsitektur awal di *localhost*. Tujuan dilakukan tahapan ini adalah agar pengembangan dan perancangan dilakukan terlebih dahulu dengan membuat *prorotype* sebelum di-*deploy* kedalam *server* atau VPS. Kemudian membuat *database PostgreSQL*,



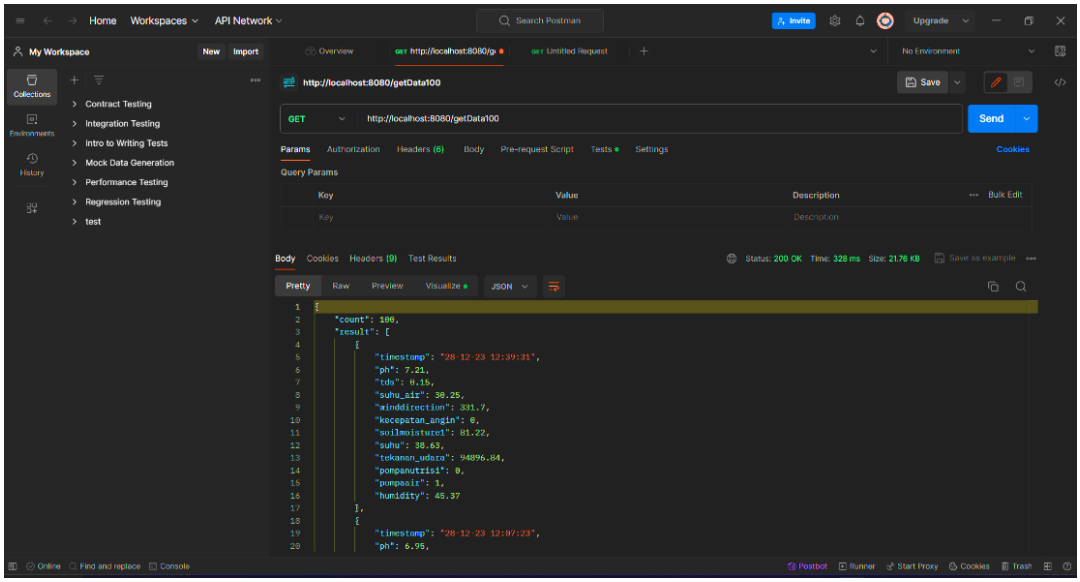
Gambar 3.5 *Subscribe* topic *MQTT* pada *software MQTTX*

Tahap selanjtnya melakukan *subscribe* ke *topic* MQTT, pada bagian ini bisa dilakukan dengan menggunakan *software* MQTTX seperti pada Gambar 3.5 untuk *checking* apakah data yang *disubscribe* sudah atau belum mengirimkan data, melakukan *handling* program pada *Node.js* untuk menerima data dari MQTT dan *insert* data kedalam *database*, kemudian membuat API menggunakan *framework express.js*, API yang akan dibuat disesuaikan dengan kebutuhan *dashboard* sehingga perlu dilakukan pengolahan data yang ada di *database* menggunakan

controller yang merupakan salah satu fungsional *express.js*, setelah itu *routing* dari *controller* yang telah dirancang tadi akan dibuat dan dijadikan sebuah URL atau *endpoint (Rest-API).*

3.4.1.3 *Prototype Testing*

Prototype yang dibuat kemudian diuji. Pengujian ini dapat melibatkan penggunaan data *dummy* untuk memverifikasi fungsionalitas API. Jika API tidak memenuhi kriteria, proses iteratif pengujian dan perbaikan dilakukan. Setelah itu, program diuji lagi dengan data aktual dari ICS & *Climate* untuk memastikan semua berjalan seperti yang diharapkan sebelum melanjutkan ke *deployment*. Pada Gambar 3.3 bagian *prototype testing* ditandai dengan kolom berwarna merah atau kolom C.



Gambar 3.6 *Testing* API

Pada tahap ini peneliti melakukan testing terhadap program yang telah dibuat dengan mengecek apakah ada *error* saat program dijalankan, dan mengecek API yang akan diberikan pada *front-end* apakah sudah dapat menerima data dan mengirimkanya dengan baik. Pada tahap ini peneliti menggunakan *software postman. Postman* adalah *platform* pengembangan API yang populer yang memungkinkan pengembang untuk merancang, menguji, mendokumentasikan, memantau, dan mempublikasikan API mereka dengan mudah. Aplikasi ini

menyediakan antarmuka pengguna yang ramah untuk mempermudah interaksi dengan API tanpa perlu menulis kode. *Postman* terdapat fungsi *GET, PUT, POST* dan *DELETE* untuk menguji sebuah *endpoint* pada *backend* tersebut dapat berjalan sebagaimana yang diharapkan .

3.4.1.4 *Deployment Sistem*

Jika *prototype* yang dirancang sudah berjalan dengan baik dan mendapatkan hasil *testing* yang diinginkan, maka tahap selanjutnya adalah *deployment* sistem. Pada tahapan ini *prototype* yang telah dibuat akan di-*deploy* kedalam *server*, yang pada penelitian ini merupakan sebuah VPS atau *virtual private server*, bagian ini ditandai dengan kolom berwarna ungu atau kolom D pada Gambar 3.4. tahapan ini peneliti melakukan inisialisasi dan persiapan pada VPS, kemudian program *prototype* yang telah dirancang pada *localhost*, selanjutnya dilakukan *insert* data kedalam *database* yang berada di VPS, dan dilanjutkan dengan membuat API. Tahapan ini dilakukan jika tahapan pengujian telah sesuai sehingga program *prorotype* akan diupload ke *server* agar bisa diakses secara *online*.

3.4.1.5 Sistem *Testing*

Sistem *testing* merupakan tahapan yang dilakukan setelah sistem berhasil di-*deploy* kedalam *server*, pada tahapan ini hal yang dilakukan tidak jauh berbeda dengan testing *prototype* yaitu melakukan *testing* terhadap program dan API yang telah dirancang pada *prototype* apakah setelah *dideploy* kedalam *server* terdapat masalah atau tidak, perbedaan *testing prototype* dan *testing* sistem terletak pada pengujian terhadap sistem telah dilakukan menggunakan API dan program yang *online* atau berada pada *server* , sedangkan pada *testing prototype*, *testing* dilakukan pada API yang masi berjalan di arsitektur *local*. Dan pengujian dilakukan menggunakan data *realtime* yang dikirimakan oleh sensor.

3.4.1.6 Evaluasi Sistem

Evaluasi sitem merupakan tindakan yang akan dilakukan apabila dalam tahap sistem *testing* hasil yang diperoleh masih belum sesuai dengan harapan, pada

penelitian ini tahap evaluasi & *maintenance* akan dilakukan apabila terjadi *error* pada program atau API yang dibuat tidak bekerja. Sehingga akan dilakukan tahap *maintenance* dan perbaikan terhadap sistem yang telah dibangun.

3.4.1.7 Sistem Digunakan

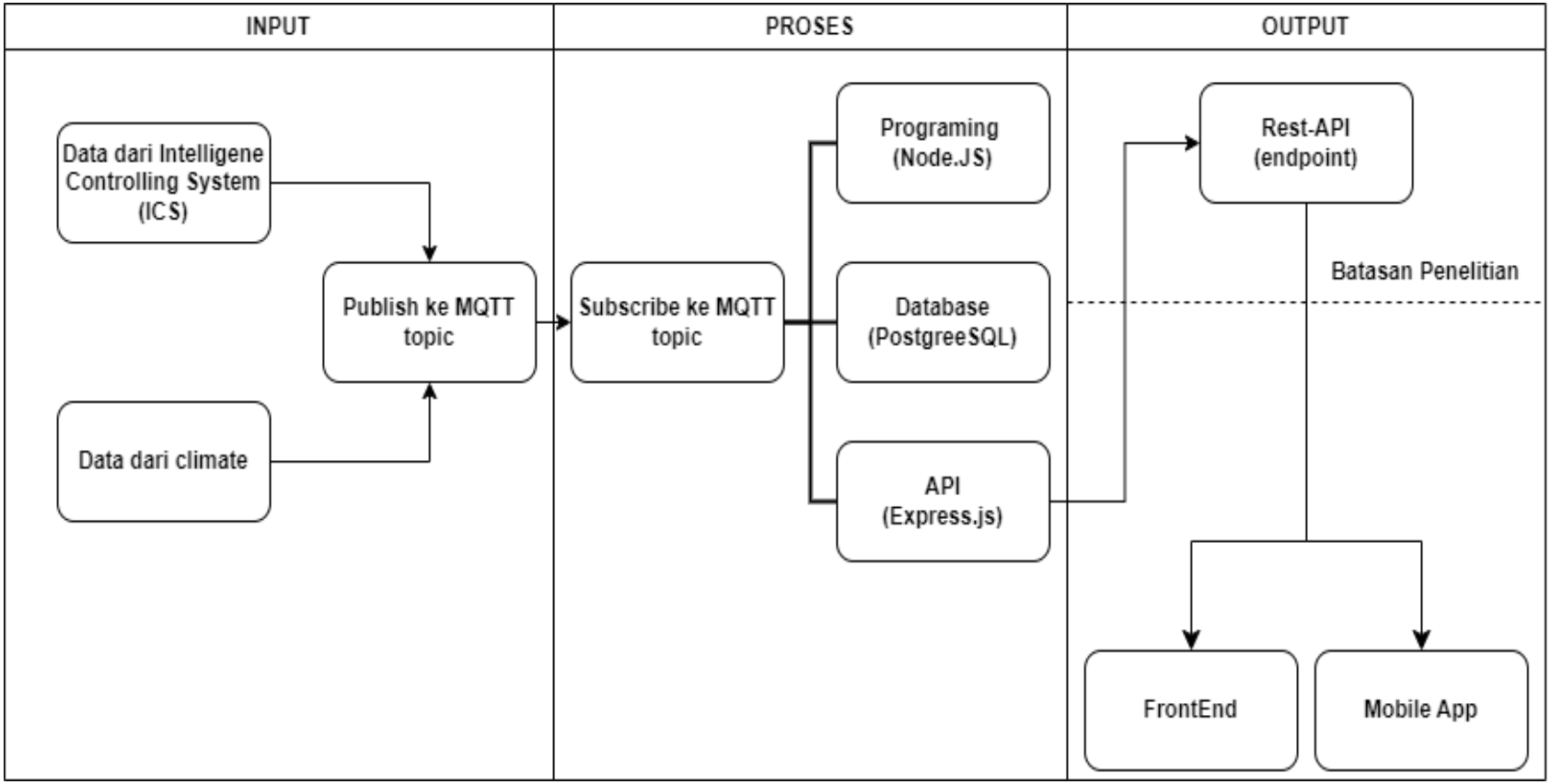
Setelah tahap pengujian selesai dan sistem dinyatakan stabil, API yang dibuat kemudian diintegrasikan dengan *frontend* dan aplikasi *mobile*, menandai sistem tersebut siap untuk digunakan dalam pengaturan operasional nyata. Ini menandakan berakhirnya siklus pengembangan dengan metodologi SDLC *prototype* dan transisi ke fase pemeliharaan dan penggunaan rutin.

3.4.2 Analisa Hasil

Tahap analisis dilakukan untuk pengujian mengenai hasil pengembangan dan implementasi *backend dashboard iot* pada *inteliggence control system (ICS) smart greenhouse* menggunakan protokol *MQTT* yang dilakukan setelah proses pengembangan telah selesai dan data hasil pengujian telah terkumpul.

Pada penelitian ini dirancang beberapa Api yang memiliki fungsi sebagai berikut :

1. *getlatestdata* : mendapatkan data terbaru dari *database*
2. *getdatafortable* : mengambil data untuk ditampilkan pada *table dashboard*
3. *getdatachart* : mengambil data untuk ditampilkan pada grafik *dashboard*
4. *getonedaydata* : mengambil data selama satu hari
5. *getoneweekdata* : mengambil data selama satu minggu
6. *getonemonthdata* : mengambil data selama satu bulan
7. *download100data* : download 100 data terbaru
8. *downloaddailydata* : download data selama satu hari
9. *downloadweeklydata* : download data selama satu minggu
10. *downloadmonthlydata* : download data selama satu bulan



Gambar 3.7 *Input* dan *Output* pada penelitian yang dilakukan

Seperti yang diilustrasikan pada Gambar 3.7, analisa hasil didasarkan pada tahapan *input* yang diterima dan proses yang dikerjakan sehingga menghasilkan sebuah *output* yang diharapkan pada pengembangan sistem ini. Pada tahap *input*, sistem menerima data dari *Intelligent Controlling System* (ICS) dan *climate* yang secara *real-time mempublish* data tersebut ke topik MQTT. Ini memungkinkan data untuk dikomunikasikan melalui protokol MQTT yang efisien dan cepat. Setelah data *dipublish*, *backend* kemudian *subscribe* ke topik MQTT yang sama untuk mengambil data yang dikirim.

Dengan *Node.js*, data yang diterima diproses dan kemudian disimpan dalam *database PostgreSQL*, yang merupakan basis data yang dapat menangani transaksi data besar dengan keamanan yang tinggi. Setelah data tersimpan, sistem menggunakan *Express.js*, sebuah *framework* populer dalam ekosistem *Node.js*, untuk membangun API yang melayani sebagai jembatan antara *database* dan aplikasi pengguna.

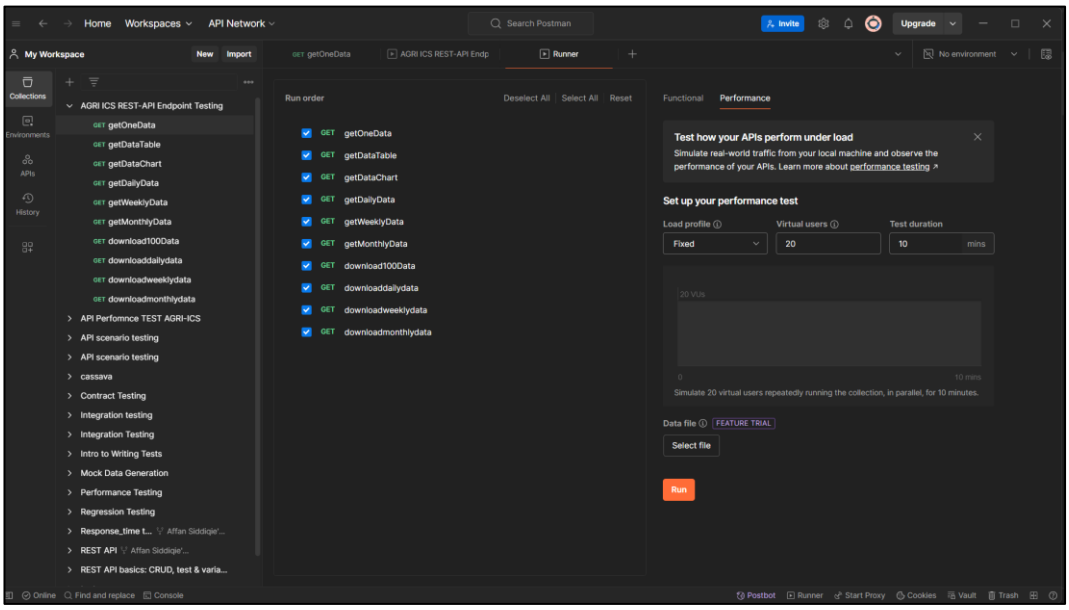
Output dari proses ini adalah sebuah *Rest-API* yang menyediakan *endpoint* untuk *frontend* dan aplikasi *mobile*. *Endpoint* ini memungkinkan sistem *frontend* untuk mengambil data yang dibutuhkan dan aplikasi *mobile* untuk mengintegrasikan data tersebut ke dalam pengalaman pengguna *mobile* yang interaktif. Dengan cara ini, data yang dikumpulkan dan diproses oleh sistem *backend* dengan aman dan efisien tersedia untuk aplikasi klien, memastikan bahwa pengguna akhir dapat mengakses informasi terbaru dengan cepat dan mudah.

3.4.2.1 Analisa Kinerja API

Tahapan yang dilakukan terhadap *Rest-Api* yang telah dirancang, berfungsi untuk menilai kemampuan dan kinerja sebuah API. Pada pengujian ini penulis akan melakukan sejumlah *test* terhadap API seperti *Performance Testing* dan *Throughput Testing* Berdasarkan hasil analisis dapat dinilai apakah API yang dirancang memiliki kinerja dan kemapuan yang baik atau tidak. Dan hasil analisis ini dapat dijadikan bahan evaluasi dalam pengembangan yang penulis lakukan.

3.4.2.1.a *Performance Testing*

Tahapan yang dilakukan pada API *testing* adalah *performance testing*. *Testing* ini dilakukan untuk melihat kemampuan setiap *endpoint REST-API* untuk memberikan respon dan melihat Tingkat keberhasilan dengan melihat persentase *error rate dan respon time* pada pengujian.



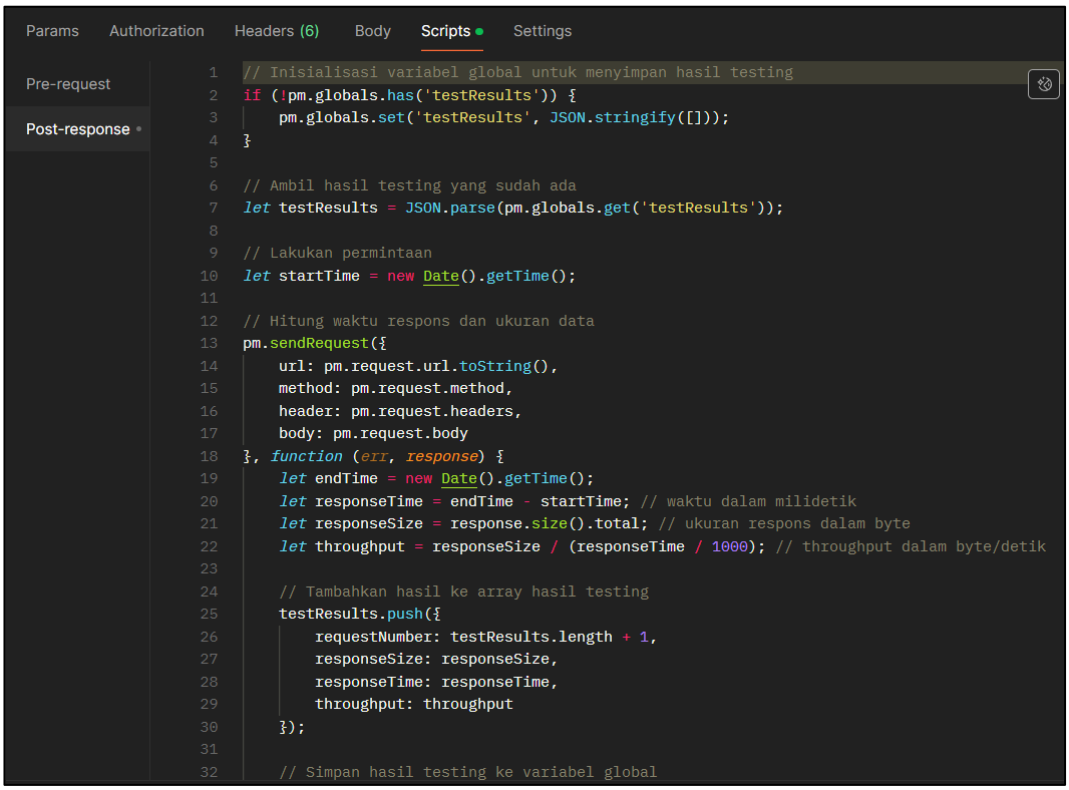
Gambar 3.8 *Runner Postman Performance Testing*

Performance testing bertujuan untuk mengevaluasi kemampuan dan kinerja setiap *endpoint REST-API* dalam kondisi beban tertentu. *Testing* ini dilakukan dengan cara mengirimkan sejumlah permintaan ke *endpoint* dan mengukur *respon time* serta persentase *error rate* yang terjadi. Dengan melakukan *performance testing*, dapat diketahui sejauh mana API dapat menangani beban kerja yang tinggi dan mengidentifikasi *bottleneck* yang mungkin ada dalam sistem. Hal ini sangat penting untuk memastikan bahwa program dapat berfungsi dengan baik dalam kondisi sebenarnya dan memberikan pengalaman pengguna yang optimal.

3.4.2.1.b *Throughput Testing*

Throughput testing dilakukan untuk mengetahui jumlah data yang diproses oleh sebuah API dalam jangka waktu tertentu. Pengujian *throughput* juga dilakukan untuk mengetahui pengaruh terhadap nilai *respon time* dan *error rate* yang muncul

pada *performance testing* yang telah diuji sebelumnya.




```
// Inisialisasi variabel global untuk menyimpan hasil testing
if (!pm.globals.has('testResults')) {
    pm.globals.set('testResults', JSON.stringify([]));
}

// Ambil hasil testing yang sudah ada
let testResults = JSON.parse(pm.globals.get('testResults'));

// Lakukan permintaan
let startTime = new Date().getTime();

// Hitung waktu respons dan ukuran data
pm.sendRequest({
    url: pm.request.url.toString(),
    method: pm.request.method,
    header: pm.request.headers,
    body: pm.request.body
}, function (err, response) {
    let endTime = new Date().getTime();
    let responseTime = endTime - startTime; // waktu dalam milidetik
    let responseSize = response.size().total; // ukuran respons dalam byte
    let throughput = responseSize / (responseTime / 1000); // throughput dalam byte/detik

    // Tambahkan hasil ke array hasil testing
    testResults.push({
        requestNumber: testResults.length + 1,
        responseSize: responseSize,
        responseTime: responseTime,
        throughput: throughput
    });

    // Simpan hasil testing ke variabel global
```

Gambar 3.9 Konfigurasi *Postman Throughput Testing*

Pengujian *throughput* pada *postman* dilakukan menggunakan program yang telah disesuaikan dengan perhitungan *throughput* yaitu.

$$throughput = \frac{respon\ size}{respon\ time}$$

Berdasarkan konfigurasi *throughput* pada Gambar 3.9 maka akan dihasilkan 20 nilai pengujian *throughput testing* dengan rumus yang telah ditentukan dan hasil *throughput* akan ditampilkan dalam satuan *byte*/detik.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan penelitian dan analisa yang telah dilakukan, didapat kesimpulan-kesimpulan sebagai berikut :

1. Data sensor pada *AGRI-ICS* dapat diterima dan dikirmkan menggunakan *protocol* MQTT menggunakan *broker* VPS dan data dapat disimpan dan diolah pada *database*.
2. Program *backend* yang dirancang dapat berjalan dengan baik pada proses pengembangan terdapat beberapa *error* tetapi dapat di *maintenance* dengan baik. Serta telah menghasilkan *REST-API* yang digunakan pada proses pengembangan *dashboard* AGRI-ICS.
3. Data sensor yang telah tersimpan di dalam *database* VPS dapat diambil kembali menggunakan *REST-API* dan digunakan oleh tim *front-end* sehingga data tersebut dapat ditampilkan pada *dashboard monitoring* AGRI-ICS.
4. Berdasarkan hasil pengujian performa API yang telah dilakukan terhadap *endpoint getOneData,getdatatable*, dan *getdatachart* yang merupakan API dengan frekuensi permintaan paling tinggi diperoleh *respon time* dengan rata-rata 286 ms dan *error rate* 0.01% yang berarti API memiliki respon yang sangat baik karena dibawah 1 detik atau 1000ms dan kemungkinan *error* yang sangat kecil. Hasil ini menunjukan bahwa API yang telah digunakan memiliki performa yang sesuai dengan harapan.
5. Berdasarkan pengujian *throughput* Secara keseluruhan, menunjukkan bahwa API mampu menangani beban permintaan dengan baik .Pada *endpoint getdataforonemonth* API memiliki *respon time,size,dan throughput* yang lebih tinggi dikarenakan kompleksitas *query* yang dilakukan pada proses *endpoint* ini lebih rumit dan data yang lebih besar, namun masih memenuhi standar kebutuhan.

5.2 Saran

Terdapat beberapa saran untuk dilakukan pada penelitian selanjutnya antara lain adalah sebagai berikut :

1. Data sensor yang tersimpan dapat diolah dan digunakan kembali sebagai bahan penelitian lebih lanjut.
2. Penggunaan *query-query* pada beberapa *endpoint* cukup kompleks sehingga sedikit membuat proses pengolahan data sedikit lebih lama, untuk penelitian yang akan datang dapat mencari formula yang dapat meringankan kinerja API.

DAFTAR PUSTAKA

[1] D. Kurniawati and S. Rahayu, “Tantangan dan Strategi Petani Kecil di Indonesia Dalam Menghadapi Ketidakpastian Cuaca,” Jurnal Agribisnis Indonesia, vol. 10, no. 1, pp. 73-85, 2022.

[2] S. Aminah and J. Sujono, “Dinamika Pemanfaatan Lahan dan Dampaknya pada Lahan Pertanian di Indonesia,” Jurnal Agroekonomi, vol. 37, no. 2, pp. 175-188, 2019.

[3] S. Few, “Information Dashboard Design: Displaying Data for At-a-Glance Monitoring”. Analytics Press, 2013.

[4] E. Anthony and Tony, "Perancangan Aplikasi Manajemen Data Publikasi dan Penelitian," Jurnal Ilmu Komputer dan Sistem Informasi, vol. 1, pp. 1-10, 2020.

[5] N. A-hussein and A. D. Salman, “IoT Monitoring System Based on MQTT Publisher/Subscriber Protocol,” “Iraqi Journal of Computers, Communications, Control and Systems Engineering”, no. April, pp. 75–83, 2020, doi: 10.33103/uot.ijccce.20.3.7.

[6] D. Hermanto, A. N. Salim, and I. P. Putra, “Sistem Monitoring Suhu Dan Kelembapan Udara Menggunakan Protokol Mqtt Berbasis Wemos D1 Mini,” “AVoER (Applicable Innovations of Engineering and Science Research)”, pp. 27–28, 2021.

[7] H. P. Safitri and R.K. Putro, “Implementasi REST API untuk Komunikasi Antara ReactJS dan NodeJS (Studi Kasus: Modul Manajemen User Solusi247),” “Automata”, vol. 2, no. 1, pp. 0–4, 2021.

[8] D. R. Agustina, A. Y. Vandika, W. Susanty, T. Tanjung, and R. N. Afiani, “Implementasi Service Data untuk Pemantauan Lighting pada Smart Agriculture,” “Digital Transformation Technology”, vol. 3, no. 2, pp. 380–388, 2023.

[9] M. S. Tahir and A. Rahim, "The Role of Smart Greenhouses in Precision Agriculture: A Review," in “Proc. Int. Conf. on Advanced Technologies for Agricultural and Environmental Engineering (ATAGEE)”, 2021, pp. 134-139.

[10] S. Mulyono, M. Taufik, and M. Taufiqurrohman, “Sistem IoT Terintegrasi Menggunakan Flow Based Programming dengan Protokol MQTT dan Time Series DB,” “Jurnal Transistor Elektro dan Informatika”, vol. 3, no. 1, pp. 9–20, 2018.

[11] "MQTT (The Standard for IoT Messaging)," MQTT.org. [Online]. Available: https://mqtt.org/. [Diakses: Dec. 20, 2023].

[12] "Eclipse Mosquitto," Eclipse. [Online]. Available: https://mosquitto.org/. [Diakses: Dec. 25, 2023].

[13] R. P. Eka, A. Rachman, and T. H. Wahyu, “Virtual Private Server (VPS) Sebagai Alternatif Pengganti Dedicated Server,” in “11th Seminar on Intelligent Technology and Its Applications, SITIA”, 2010. [Online]. Available: http://www.apnicservices.com/. [Diakses: Feb. 9, 2024].

[14] C. Y. Andika and S. Rudiarto, “Rancang Bangun Aplikasi Sosial Media Crawler Menggunakan Nodejs Menerapkan Konsep Non-Blocking I/O,” “Jurnal Ilmiah FIFO”, vol. IX, no. 2, 2017.

[15] A. Hidayatullah and B. Purwanto, "Pembangunan Web Service menggunakan Framework Express.js pada Node.js," “Jurnal Teknik Informatika”, vol. 12, no. 1, pp. 34-39, 2020.

[16] "PostgreSQL," PostgreSQL.org. [Online]. Available: https://www.postgresql.org/about/. [Diakses: Dec. 20, 2023].

[17] M. F. A. Muri, H. S. Utomo, and R. Sayyidati, “Search Engine Get Application Programming Interface,” “Jurnal Sains dan Informatika”, vol. 5, no. 2, pp. 88–97, Dec. 2019, doi: 10.34128/jsi.v5i2.175.

[18] Y. K. Kolondam, "Mengenal Metode SDLC dalam Pengembangan Software," Gamelab Indonesia, Jan. 11, 2022. [Online]. Available: https://www.gamelab.id/news/1345-mengenal-metode-sdlc-dalam-pengembangan-software. [Diakses: Feb. 9, 2024].

[19] M. A. P. Domingues, "Performance testing of open-source HTTP web frameworks in an API," in “Proceedings of the 12th Doctoral Symposium in Informatics Engineering - DSIE'17”, Porto, Portugal, 2017, pp. 8-15.

[20] J. Berg and D. Mebrahtu Redi, "Benchmarking the request throughput of conventional API calls and gRPC: A Comparative Study of REST and gRPC," B.S. thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2023.

[21] TatvaSoft, "MQTT Publish Subscribe Architecture," 2021. [Online]. Available: https://www.tatvasoft.com/blog/wp-content/uploads/2021/10/MQTT-Publish-Subscribe-Architecture.jpg. [Diakses: Feb. 9, 2024].

[22] Mosquitto, "Mosquitto Broker," 2024. [Online]. Available: https://images.app.goo.gl/FBdFPxsm7se2prMC6. [Diakses: Feb. 9, 2024].

[23] IronTree, "VPS Graphic," 2019. [Online]. Available: https://www.irontree.co.za/wp-content/uploads/2019/06/VPS-Graphic-1024x768.png. [Diakses: Feb. 9, 2024].

[24] Wikimedia Commons, "Node.js Logo," 2024. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/d/d9/Node.js_logo.svg. [Diakses: Feb. 9, 2024].

[25] Medium, "Image related to Express.js," 2024. [Online]. Available: https://miro.medium.com/v2/resize:fit:1400/1”i2fRBk3GsYLeUk_Rh7AzHw.png. [Diakses: Feb. 9, 2024].

[26] ITBox.id, "Image related to PostgreSQL," 2023. [Online]. Available: https://itbox.id/wpcontent/uploads/2023/02/1_7AOhGDnRL2eyJMUidCHZEA.jpeg. [Diakses: Feb. 9, 2024].

[27] Binar, "Software Development Life Cycle," 2024. [Online]. Available: https://assets-global.website-files.com/6100d0111a4ed76bc1b9fd54/62e3a7dc35f14275f8bbd7ac_B_E_BVzm58Vcl-x1K6XbsvLXf6GLsZQ1-tdzw9CApnEmWEGdNgSHfXAeY-ihWL_1CMY-nPXSlz-x-HACttTPaZQc8rWnmbo45N2nISCpbSumeNPiQbu1YaiqisUElPMf9fK2NANFBNkP7DHw6sYajw.png. [Diakses: Feb. 9, 2024].

[28] S. S. Raweyai and I. R. Widiasari, "Performance Testing of Academic Website Using Load Testing Method Supported by Apache JMeterTM at XYZ University," “Jurnal Teknik Informatika (JTIF)”, vol. 5, no. 3, pp. 721-730, Jun. 2024.

[29] O. Nilsson and N. Yngwe, "API Latency and User Experience: What Aspects Impact Latency and What are the Implications for Company Performance?," B.S. thesis, Skolan för Elektroteknik och Datavetenskap, KTH Royal Institute of Technology, Stockholm, Sweden, 2022.