

**PENGEMBANGAN WEB DASHBOARD UNTUK
PEMANTAUAN KONDISI LINGKUNGAN AKUAKULTUR
YANG TERISOLASI DARI JARINGAN INTERNET DENGAN
PENGIRIMAN DATA MENGGUNAKAN MODUL LORA**

TUGAS AKHIR

**Karya tulis sebagai salah satu syarat
untuk memperoleh gelar Sarjana dari
Institut Teknologi Bandung**

Oleh
MUHAMMAD SHAFLY NURRASYID
NIM: 18120076
(Program Studi Teknik Telekomunikasi)



INSTITUT TEKNOLOGI BANDUNG
Agustus 2024

ABSTRAK

PENGEMBANGAN WEB DASHBOARD UNTUK PEMANTAUAN KONDISI LINGKUNGAN AKUAKULTUR YANG TERISOLASI DARI JARINGAN INTERNET DENGAN PENGIRIMAN DATA MENGGUNAKAN MODUL LORA

Oleh
Muhammad Shafly Nurrasyid
NIM: 18120076
(Program Studi Teknik Telekomunikasi)

Penelitian ini mengembangkan sebuah *web dashboard* untuk memantau kondisi lingkungan akuakultur yang terisolasi dari jaringan internet, dengan memanfaatkan modul LoRa (*Long Range*) untuk pengiriman data. Tantangan utama dalam pengelolaan akuakultur di daerah terpencil adalah keterbatasan akses internet, yang menghambat pengiriman data dari jaringan sensor IoT (*Internet of Things*) ke server. LoRa dipilih sebagai solusi karena kemampuannya untuk mengirim data jarak jauh dengan konsumsi daya rendah (*Low Power Wide Area Network, LPWAN*), menjadikannya solusi ideal untuk daerah yang sulit dijangkau.

Dalam implementasinya, sensor IoT yang digunakan untuk memantau berbagai parameter lingkungan, seperti suhu air, pH, dan kadar oksigen, dihubungkan dengan modul LoRa melalui mikrokontroler seperti Arduino dan ESP32. Data yang dikumpulkan oleh sensor kemudian dikirimkan melalui jaringan LoRa ke LoRaWAN *gateway*, yang terhubung dengan jaringan berbasis IP (*Internet Protocol*). Dari LoRaWAN *gateway*, data dikirimkan dan disimpan di server milik penyedia layanan LoRa, seperti Antares Telkom. Data ini kemudian diakses oleh *web dashboard* yang dikembangkan, yang memungkinkan pengelola akuakultur untuk memantau kondisi lingkungan secara *real-time* dan membuat keputusan yang lebih baik.

Pada bagian *frontend*, pengembangan *web dashboard* dilakukan dengan menggunakan HTML (*Hyper Text Markup Language*) untuk menyusun struktur halaman web, sementara CSS (*Cascading Style Sheets*) dan *framework* Bootstrap digunakan untuk *styling* visual, sehingga memberikan tampilan yang menarik dan responsif. JavaScript digunakan untuk menambah interaktivitas, sehingga membuat halaman web lebih dinamis. Pada bagian *backend*, *framework* Flask digunakan untuk menangani logika aplikasi web dan menyediakan API yang mengelola permintaan dari *frontend*. Sistem manajemen basis data MySQL digunakan untuk menyimpan data, dengan SQLAlchemy sebagai alat komunikasi antara Flask dan MySQL. Data dari sensor IoT yang disimpan di server Antares diambil melalui integrasi antara *backend* Flask dengan API Antares, kemudian data tersebut didekompresi menggunakan *library* Unishox2 sebelum disimpan di MySQL.

Analisis hasil pengujian *user interface* menunjukkan bahwa desain *frontend web dashboard* mendapatkan nilai indeks skala Likert yang tinggi, dengan nilai rata-rata di atas 87%, yang menunjukkan kemudahan navigasi pada dashboard, kemudahan pemahaman informasi dari data yang ditampilkan, kecepatan pemuatannya pada dashboard, dan kecocokan dashboard untuk pemantauan data sensor IoT. Pada bagian *backend*, pengujian kinerja *web dashboard* menunjukkan bahwa semua menu, submenu, dan fitur berfungsi dengan latensi yang bervariasi antara 30 ms hingga 133 ms. Nilai latensi tersebut masih dapat diterima untuk aplikasi web pemantauan data. Latensi tertinggi terjadi pada submenu dashboard karena terdapat banyak proses pengambilan dan pengecekan data dari database. Pada bagian *backend* juga dilakukan pengujian untuk membuktikan bahwa data dari Antares berhasil di-*backup* dan disimpan pada database MySQL secara otomatis setiap kali user mengakses submenu dashboard.

Kata kunci: *akuakultur, sensor IoT, web dashboard, pemantauan data, LoRa, MySQL, Flask.*

ABSTRACT

DEVELOPMENT OF A WEB DASHBOARD FOR MONITORING AQUACULTURE ENVIRONMENTAL CONDITIONS ISOLATED FROM THE INTERNET NETWORK USING LORA MODULE DATA TRANSMISSION

By

Muhammad Shafly Nurrasyid

NIM: 18120076

(Telecommunication Engineering Program)

This research develops a web dashboard to monitor environmental conditions in aquaculture that are isolated from the internet network, utilizing LoRa (Long Range) modules for data transmission. The main challenge in managing aquaculture in remote areas is the limited internet access, which hinders data transmission from IoT (Internet of Things) sensor networks to servers. LoRa is chosen as a solution due to its capability to transmit data over long distances with low power consumption (Low Power Wide Area Network, LPWAN), making it an ideal solution for hard-to-reach areas.

In the implementation, IoT sensors used to monitor various environmental parameters, such as water temperature, pH, and oxygen levels, are connected to LoRa modules via microcontrollers like Arduino and ESP32. The data collected by the sensors is then transmitted via the LoRa network to a LoRaWAN gateway, which is connected to an IP-based (Internet Protocol) network. From the LoRaWAN gateway, the data is sent and stored on a LoRa service provider's server, such as Antares Telkom. This data is then accessed by the developed web dashboard, allowing aquaculture managers to monitor environmental conditions in real-time and make better decisions.

On the frontend, the web dashboard is developed using HTML (HyperText Markup Language) to structure the web pages, while CSS (Cascading Style Sheets) and the Bootstrap framework are used for visual styling, providing an attractive and responsive appearance. JavaScript is employed to add interactivity, making the web pages more dynamic. On the backend, the Flask framework is used to handle web application logic and provide an API that manages requests from the frontend. MySQL database management system is used for data storage, with SQLAlchemy as the tool for communication between Flask and MySQL. Data from IoT sensors stored on Antares servers is accessed through integration between Flask backend and the Antares API, then decompressed using the Unishox2 library before being stored in MySQL.

Analysis of the user interface testing results shows that the frontend design of the web dashboard received a high Likert scale index value, with an average score

above 87%, indicating ease of navigation on the dashboard, ease of understanding information from the displayed data, data loading speed on the dashboard, and suitability of the dashboard for monitoring IoT sensor data. On the backend, the performance testing of the web dashboard shows that all menus, submenus, and features function with latencies ranging from 30 ms to 133 ms. This latency is still acceptable for data monitoring web applications. The highest latency occurs in the dashboard submenu due to the numerous processes involved in fetching and checking data from the database. Backend testing also verifies that data from Antares is successfully backed up and stored in the MySQL database automatically each time a user accesses the dashboard submenu.

Keywords: aquaculture, IoT sensors, web dashboard, data monitoring, LoRa, MySQL, Flask.

**PENGEMBANGAN WEB DASHBOARD UNTUK
PEMANTAUAN KONDISI LINGKUNGAN AKUAKULTUR
YANG TERISOLASI DARI JARINGAN INTERNET DENGAN
PENGIRIMAN DATA MENGGUNAKAN MODUL LORA**

HALAMAN PERSETUJUAN

Oleh
Muhammad Shafly Nurrasyid
NIM: 18120076
(Program Studi Teknik Telekomunikasi)

Institut Teknologi Bandung

Menyetujui
Tim Pembimbing

Tanggal 29 Juli 2024

Ketua

Anggota



(Aldo Agusdian, S.T, M.T.)



(Dr. Iskandar, S.T, M.T.)

**PENGEMBANGAN WEB DASHBOARD UNTUK
PEMANTAUAN KONDISI LINGKUNGAN AKUAKULTUR
YANG TERISOLASI DARI JARINGAN INTERNET DENGAN
PENGIRIMAN DATA MENGGUNAKAN MODUL LORA**

HALAMAN PENGESAHAN

Oleh
Muhammad Shafly Nurrasyid
NIM: 18120076
(Program Studi Teknik Telekomunikasi)

Institut Teknologi Bandung

Menyetujui
Tim Pembimbing

Tanggal 29 Juli 2024

Ketua



(Aldo Agusdian, S.T, M.T.)

Anggota



(Dr. Iskandar, S.T, M.T.)

PEDOMAN PENGGUNAAN BUKU TUGAS AKHIR

Buku Tugas Akhir yang tidak dipublikasikan terdaftar dan tersedia di Perpustakaan Institut Teknologi Bandung, dan terbuka untuk umum dengan ketentuan bahwa hak cipta ada pada penulis dengan mengikuti aturan HaKI yang berlaku di Institut Teknologi Bandung. Referensi kepustakaan diperkenankan dicatat, tetapi pengutipan atau peringkasan hanya dapat dilakukan seizin penulis dan harus disertai dengan kaidah ilmiah untuk menyebutkan sumbernya.

Situs hasil tugas akhir ini dapat dituliskan dalam bahasa Indonesia sebagai berikut:

Nurrasyid, M. S. (2024): *Pengembangan Web Dashboard untuk Pemantauan Kondisi Lingkungan Akuakultur yang Terisolasi dari Jaringan Internet dengan Pengiriman Data Menggunakan Modul LoRa*, Tugas Akhir Program Studi Teknik Telekomunikasi, Institut Teknologi Bandung.

dan dalam bahasa Inggris sebagai berikut:

Nurrasyid, M. S. (2024): *Development of a Web Dashboard for Monitoring Aquaculture Environmental Conditions Isolated from the Internet Network Using LoRa Module Data Transmission*, Telecommunication Engineering Program, Institut Teknologi Bandung.

Memperbanyak atau menerbitkan sebagian atau seluruh buku tugas akhir haruslah seizin Dekan Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.

Dipersembahkan kepada bapak, ibu, kakak, adik, serta keluarga besar, dan teman-temanku tercinta yang senantiasa mendukung lahir dan batin.

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas berkat dan karunia-Nya yang telah memberikan kesempatan penulis untuk menyelesaikan salah satu kewajiban dalam menempuh studi sarjana S1 pada Program Studi Teknik Telekomunikasi di Institut Teknologi Bandung yaitu Tugas Akhir serta laporannya yang berjudul “ Pengembangan Dashboard Berbasis Web untuk Pemantauan Kondisi Lingkungan Akuakultur yang Terisolasi dari Jaringan Internet dengan Pengiriman Data Menggunakan Modul LoRa”.

Ucapan terima kasih dan rasa syukur penulis sampaikan kepada seluruh pihak yang telah membantu dan mendukung penulis dalam pelaksanaan Tugas Akhir, baik dalam bentuk usaha, pemikiran, waktu, maupun material. Oleh karena itu, izinkanlah penulis menyampaikan terima kasih kepada:

1. Bapak, Ibu, dan Saudara penulis, yang selalu mendukung baik dalam bentuk finansial, jasa, maupun kata penyemangat yang menghangatkan.
2. Bapak Dr. Tutun Juhana, S.T., M.T., selaku Dekan Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung.
3. Ibu Dr. Ing. Chairunnisa, S.T, M.T., selaku Ketua Program Studi Teknik Telekomunikasi Institut Teknologi Bandung.
4. Bapak Aldo Agusdian , S.T, M.T., selaku dosen pembimbing yang selalu membimbing dan memberi arahan dalam penggerjaan tugas ini.
5. Bapak Dr. Iskandar, S.T, M.T., selaku dosen pengampu mata kuliah Tugas Akhir sekaligus sebagai dosen pembimbing yang selalu membimbing dan memberi arahan dalam penggerjaan tugas ini.
6. Helmi Ahmad Khaderani, selaku partner saya dalam pelaksanaan Tugas Akhir ini yang senantiasa bekerja sama dalam menyelesaikan tugas ini.
7. Seluruh rekan yang membantu dan memberi saran dalam pelaksanaan Tugas Akhir ini.

Penulis laporan ini hanyalah manusia biasa yang tidak luput dari kesalahan. Oleh karena itu, penulis terbuka menerima kritik, saran, dan diskusi sebagai bahan perbaikan dan pembelajaran agar penulis dapat menjadi pribadi yang lebih baik di

masa depan. Semoga laporan Tugas Akhir yang penulis buat dapat bermanfaat bagi pembaca, terutama teman-teman penggiat telekomunikasi.

Bandung, 22 Juli 2024



Muhammad Shafly Nurrasyid
NIM 18120076

DAFTAR ISI

Abstrak	i
Abstract	iii
Halaman Persetujuan.....	v
Halaman Pengesahan	vi
Pedoman Penggunaan Buku Tugas Akhir	vii
Kata Pengantar	ix
Daftar Isi.....	xi
Daftar Gambar dan Ilustrasi	xiii
Daftar Tabel	xv
Daftar Singkatan dan Lambang.....	xvi
Bab I Pendahuluan	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah.....	2
1.3 Tujuan dan Manfaat	3
1.4 Lingkup Permasalahan.....	3
1.5 Asumsi-asumsi	3
1.6 Hipotesis	4
1.7 Sistematika Buku Tugas Akhir	4
Bab II Proses dan Pengembangan Desain	5
2.1 Tinjauan Pustaka	5
2.1.1 Teknologi LoRa dan LoRaWAN	5
2.1.2 Kompresi dan Dekompresi Data	7
2.1.3 <i>Database Management System</i>	9
2.1.3 Dashboard Berbasis Web	11
2.2 Persyaratan Desain	12
2.3 Konsep Desain	17
2.4 Pengembangan Desain	18
2.4.1 Alternatif Desain Frontend.....	18
2.4.2 Alternatif Desain Backend	20
2.4.3 Alternatif Database	21
2.4.4 Pemilihan Desain	22
Bab III Detil Desain dan Implementasi.....	23
3.1 Model Desain	23
3.1.1 Desain Frontend	23
3.1.2 Desain Backend.....	25
3.1.3 Desain Database	25
3.2 Implementasi.....	26
3.2.1 Implementasi Frontend	26
3.2.2 Implementasi Backend	31
Bab IV Pengujian dan Analisis	40
4.1 Pengujian Desain	40
4.1.1 Pengujian Desain Frontend	40

4.1.2 Pengujian Desain Backend.....	42
4.2 Analisis Hasil Pengujian	54
4.2.1 Analisis Hasil Pengujian Desain Frontend.....	54
4.2.2 Analisis Hasil Pengujian Desain Backend	56
Bab V Simpulan dan Saran	58
Daftar Pustaka	60
Lampiran A Tampilan Web Dashboard	64
Lampiran B Hasil Kuesioner Pengujian Frontend	65
Lampiran C Kode Program Frontend.....	66
Lampiran D Kode Program Backend Flask App	71
Lampiran E Kode Program Backend Database.....	78
Lampiran F Kode Program Backend Fetch Data Antares.....	85
Lampiran G Flowchart Menu dan Fitur	87

DAFTAR GAMBAR DAN ILUSTRASI

Gambar II.1 Format Paket LoRa.....	6
Gambar II.2 Susunan Protkol LoRaWAN	6
Gambar II.3 Komponen Sistem Kompresi Data	8
Gambar II.4 Pohon Objektif.....	13
Gambar II.5 Flowchart Sistem Terintegrasi.....	16
Gambar III.1 Arsitektur Frontend	23
Gambar III.2 Flowchart Navigasi Web Dashboard	24
Gambar III.3 Arsitektur Backend.....	25
Gambar III.4 Entity Relationship Diagram.....	26
Gambar III.5 Menu Home.....	27
Gambar III.6 Menu Login	27
Gambar III.7 Menu Register	28
Gambar III.8 Submenu Device	28
Gambar III.9 Submenu Dashboard	29
Gambar III.10 Submenu Data	30
Gambar III.11 Submenu Profile	30
Gambar III.12 Submenu Admin Devices.....	31
Gambar III.13 Submenu Admin Users	31
Gambar III.14 Kode Program Inisiasi Aplikasi Flask.....	32
Gambar III.15 Kode Program Pendefinisian Endpoint dan Routing	32
Gambar III.16 Kode Program Middleware	34
Gambar III.17 Kode Program Error Handling	34
Gambar III.18 Kode Program Koneksi dan Inisialisasi Database.....	35
Gambar III.19 Kode Program Pendefinisian Model Data.....	35
Gambar III.20 Kode Program Fungsi CRUD	36
Gambar III.21 Kode Program Fungsi Query	36
Gambar III.22 Kode Program Fetch Data API Antares	37
Gambar III.23 Kode Program Dekompresi	38
Gambar III.24 Kode Program Store Data	38
Gambar IV.1 Pengujian Menu Home	43
Gambar IV.2 Pengujian Menu Login.....	44
Gambar IV.3 Pengujian Menu Register	44
Gambar IV.4 Pengujian Submenu Device	45
Gambar IV.5 Pengujian Submenu Dashboard	46

Gambar IV.6 Pengujian Submenu Data	46
Gambar IV.7 Pengujian Submenu Profile.....	47
Gambar IV.8 Pengujian Submenu Admin Devices	48
Gambar IV.9 Pengujian Submenu Admin Users	48
Gambar IV.10 Pengujian Fitur Logout	49
Gambar IV.11 Pengujian Fitur Download Data.....	49
Gambar IV.12 Pengujian Fitur Edit User.....	50
Gambar IV.13 Pengujian Fitur Add Device.....	51
Gambar IV.14 Pengujian Fitur Edit Device	51
Gambar IV.15 Pengujian Fitur Delete Device	52
Gambar IV.16 Pengujian Store Data.....	53
Gambar IV.17 Pengujian Backup Data.....	54

DAFTAR TABEL

Tabel II.1 Tabel <i>Bit Rate</i> dan <i>Maximum Payload</i> untuk Parameter LoRa Tertentu	7
Tabel II.2 Tabel Persyaratan Fungsional dan Fungsi.....	14
Tabel II.3 Tabel Metrik Pengukuran dan Syarat Pemenuhan	17
Tabel III.1 Pembagian Subsistem	23
Tabel III.2 Endpoint API Aplikasi Flask dan Kegunaannya.....	33
Tabel III.3 Daftar Fungsi CRUD dan Kegunaannya.....	37
Tabel IV.1 Klasifikasi Indeks Skala Likert.....	41
Tabel IV.2 Hasil Kuesioner Penilaian User Interface	42
Tabel IV.3 Analisis Hasil Kuesioner Penilaian User Interface	55
Tabel IV.4 Analisis Hasil Pengujian Kinerja Web Dashboard	56

DAFTAR SINGKATAN DAN LAMBANG

SINGKATAN	Nama	Pemakaian pertama kali pada halaman
ACID	<i>Atomicity, Consistency, Isolation, Durability</i>	21
ALOHA	<i>Additive Links On-line Hawaii Area</i>	7
API	<i>Application Programming Interface</i>	11
BW	<i>Bandwidth</i>	5
CR	<i>Coding Rate</i>	5
CRC	<i>Cyclic Redundancy Check</i>	6
CRUD	<i>Create, Read, Update, Delete</i>	36
CSS	<i>Chirp Spread Spectrum</i>	7
CSS	<i>Cascading Style Sheets</i>	12
DBMS	<i>Database Management System</i>	9
DO	<i>Dissolved Oxygen</i>	3
DOM	<i>Document Object Model</i>	19
ERD	<i>Entity Relationship Diagram</i>	3
ESP32	<i>Espressif Systems 32-bit Microcontroller</i>	2
FAO	<i>Food and Agriculture Organization</i>	1
HTML	<i>HyperText Markup Language</i>	12
HTTP	<i>HyperText Transfer Protocol</i>	11
HTTPS	<i>HyperText Transfer Protocol Secure</i>	11
IoT	<i>Internet of Things</i>	1
ISM	<i>Industrial, Scientific, and Medical</i>	7
ISO	<i>International Organization for Standardization</i>	18
ITU-T	<i>International Telecommunication Union – Telecommunication Standardization Sector</i>	18
JSON	<i>JavaScript Object Notation</i>	38
JSX	<i>JavaScript XML</i>	19
JWT	<i>JSON Web Token</i>	34
LoRa	<i>Long Range</i>	2
LoRaWAN	<i>Long Range Wide Area Network</i>	2
LPWA	<i>Low Power Wide Area</i>	
LPWAN	<i>Low Power Wide Area Network</i>	2
LZ	<i>Lempel-Ziv</i>	9
LZO	<i>Lempel-Ziv-Oberhumer</i>	8
MAC	<i>Media Access Control</i>	6
ms	<i>Millisecond</i>	18
NoSQL	<i>Not Only SQL</i>	10
ORM	<i>Object Relational Mapping</i>	20
ORP	<i>Oxidation-Reduction Potential</i>	3
PC	<i>Personal Computer</i>	11

SINGKATAN	Nama	Pemakaian pertama kali pada halaman
PHP	<i>Hypertext Preprocessor</i>	12
pH	<i>Potential of Hydrogen</i>	3
RDBMS	<i>Relational Database Management System</i>	21
SDPPI	Satuan Data dan Pengendalian Pembangunan Infrastruktur	17
SF	<i>Spreading Factor</i>	5
SNR	<i>Signal-to-Noise Ratio</i>	5
SQL	<i>Structured Query Language</i>	10
TDS	<i>Total Dissolve Solid</i>	3
ToA	<i>Time of Air</i>	5
Ts	<i>Time Symbol (Symbol Time)</i>	5
UI	<i>User Interface</i>	40
Unishox	<i>Universal Short Hand for XML</i>	9
URL	<i>Uniform Resource Locator</i>	32

LAMBANG	Nama	Pemakaian pertama kali pada halaman
P_n	Pilihan angka skor Likert	41
T	Jumlah responden yang memilih skor Likert tertentu	41
Y	Nilai maksimum total skor skala Likert	41

BAB I PENDAHULUAN

1.1 Latar Belakang

Pada tahun 2022, produksi global perikanan dan akuakultur mencapai rekor tertinggi sebesar 223,2 juta ton, yang terdiri dari 185,4 juta ton hewan air dan 37,8 juta ton alga. Produksi ini meningkat sebesar 4,4 persen dibandingkan dengan tahun 2020 yang menunjukkan pertumbuhan signifikan dalam industri perikanan dan akuakultur global. Negara-negara di Asia berperan besar dalam produksi ini dengan menyumbangkan sebesar 70 persen dari total produksi hewan air global. Indonesia sendiri menghasilkan 12,7 juta ton hewan air pada tahun yang sama, yang mewakili 7 persen dari total produksi hewan air global (FAO, 2024).

Seiring dengan meningkatnya produksi hewan air, masyarakat mulai mencari pendekatan baru dalam pengelolaan akuakultur yang lebih modern dan efisien. Salah satu inovasi yang dapat diterapkan yaitu pemanfaatan teknologi *Internet of Things* (IoT). Teknologi IoT dapat mempermudah sistem pemantauan kualitas air pada akuakultur, yang berpotensi meningkatkan efisiensi pengelolaan akuakultur serta kualitas hasil produksi hewan air. Dengan teknologi IoT, dapat dibuat sistem pemantauan yang mendeteksi dan mengontrol berbagai parameter seperti suhu, nilai pH, oksigen terlarut, dan ketinggian air. Data parameter tersebut dikumpulkan secara *real-time* dari sensor IoT dan dikirimkan ke mikrokontroler untuk diolah dan dianalisis sehingga dapat diambil tindakan yang diperlukan untuk pengelolaan akuakultur (Abinaya dkk., 2019).

Salah satu perusahaan di Indonesia yang memanfaatkan teknologi IoT untuk membantu pertumbuhan akuakultur adalah PT Multidaya Teknologi Nusantara (eFishery). Perusahaan eFishery mengembangkan perangkat cerdas untuk akuakultur, yang memungkinkan pemberian pakan ikan secara otomatis. Perangkat eFishery terhubung dengan aplikasi milik eFishery yang dapat dikontrol melalui smartphone dan laptop (Wahyuni Sabran & Zalfiana Rusfian, 2023).

Namun, dalam pengelolaan akuakultur menggunakan teknologi IoT, terdapat tantangan ketika lokasi akuakultur berada di daerah yang terisolasi dari jaringan internet. Dalam situasi ini, jaringan lokal pada akuakultur yang terdiri dari beberapa sensor IoT tidak dapat mengirimkan data ke server secara langsung karena tidak adanya koneksi internet. Untuk mengatasi tantangan tersebut, diperlukan solusi *backhaul connection* yang dapat menghubungkan jaringan lokal pada akuakultur ke jaringan internet terdekat.

Salah satu solusi yang dapat digunakan sebagai *backhaul connection* adalah teknologi modulasi LoRa (*Long Range*). LoRa memiliki kemampuan untuk komunikasi jarak jauh pada area yang luas dengan konsumsi daya yang rendah (LPWAN) (Guangyang dkk., 2022). Oleh karena itu, teknologi ini sangat cocok digunakan untuk pengiriman data sensor IoT di daerah yang terisolasi dari jaringan internet.

Sensor IoT dapat terhubung dengan LoRa melalui mikrokontroler seperti Arduino dan ESP32. Data dari sensor IoT dikirimkan melalui LoRa ke LoRaWAN Gateway yang terhubung dengan jaringan berbasis IP (*Internet Protocol*). Dari LoRaWAN Gateway, data sensor IoT dapat disimpan pada server milik provider LoRa, salah satunya Antares Telkom. Dari server Antares Telkom, perusahaan bisnis IoT seperti eFishery dapat menyimpan data sensor IoT pada server database miliknya. Setelah disimpan di server database perusahaan, data sensor IoT perlu divisualisasikan agar dapat digunakan oleh pengelola akuakultur dalam memantau kondisi lingkungan akuakultur. Oleh karena itu, diperlukan pembuatan *dashboard* untuk pemantauan kondisi lingkungan pada akuakultur yang terisolasi dari jaringan internet.

1.2 Rumusan Masalah

Bagaimana cara menampilkan data sensor IoT pada akuakultur yang terisolasi dari jaringan internet sehingga memungkinkan pemantauan kondisi lingkungan akuakultur secara *real-time* dari jarak jauh?

1.3 Tujuan dan Manfaat

Tujuan dari tugas akhir ini adalah mengembangkan desain solusi teknis untuk menampilkan data sensor IoT dari akuakultur yang terisolasi dari jaringan internet. Solusi ini memungkinkan pemantauan kondisi lingkungan akuakultur secara *real-time* dari jarak jauh.

Tugas akhir ini diharapkan memberikan manfaat dalam berbagai aspek. Secara teknis, desain ini memungkinkan pemantauan kondisi akuakultur secara *real-time* meskipun lokasi akuakultur terisolasi dari jaringan internet, sehingga pengelola yang berada jauh dari lokasi akuakultur dapat melakukan pemantauan dengan efisien. Dalam aspek sosial, desain ini membantu pengelola menjaga kesehatan organisme akuakultur, yang berdampak positif pada masyarakat yang bergantung pada sektor akuakultur. Dari aspek lingkungan, pemantauan yang efektif dapat menjaga kualitas lingkungan akuakultur dan mengurangi dampak negatif pada ekosistem sekitarnya. Secara ekonomi, solusi ini dapat meningkatkan produktivitas dan mengurangi biaya dalam pengelolaan akuakultur.

1.4 Lingkup Permasalahan

Lingkup permasalahan pada tugas akhir ini dibatasi oleh beberapa hal berikut:

1. Desain solusi untuk menampilkan data sensor IoT dibuat dalam bentuk dashboard berbasis web.
2. Server database yang digunakan untuk penyimpanan dan pengelolaan data masih bersifat sementara dan belum terintegrasi dengan sistem eFishery.
3. Dashboard berbasis web sebagai subsistem dashboard digunakan untuk memvisualisasikan data sensor yang tersimpan di server Antares. Sementara itu, pengaturan parameter pengiriman data sensor IoT dari akuakultur menggunakan LoRa ke server Antares dilakukan pada subsistem pengiriman data.

1.5 Asumsi-asumsi

Asumsi-asumsi yang digunakan dalam tugas akhir ini adalah sebagai berikut:

1. Server database yang digunakan untuk penyimpanan dan pengelolaan data dapat merepresentasikan server asli milik eFishery karena menggunakan layanan cloud.
2. Data sensor IoT yang diolah dan disimpan pada server database yaitu suhu, *potential of Hydrogen* (pH), total padatan terlarut (TDS), oksigen terlarut (DO), potensi redoks (ORP), salinitas, ketinggian air, dan kejernihan air, ditentukan berdasarkan *Entity Relationship Diagram* (ERD) milik eFishery, yang diasumsikan dapat merepresentasikan kondisi lingkungan akuakultur.

1.6 Hipotesis

Hipotesis yang diusulkan oleh penulis tugas akhir ini adalah:

Dengan menggunakan Flask sebagai *framework* desain web dan MySQL sebagai *database management system*, dapat dikembangkan *dashboard* berbasis web untuk melakukan pemantauan data sensor IoT pada akuakultur yang terisolasi dari jaringan internet secara *real-time* dari jarak jauh.

1.7 Sistematika Buku Tugas Akhir

Buku tugas akhir ini terdiri dari lima bab. Isi dari setiap bab akan menjelaskan hal-hal berikut:

1. Bab I – Pendahuluan

Bab I berisi latar belakang, rumusan masalah, tujuan dan manfaat, lingkup permasalahan, asumsi-asumsi, dan hipotesis penelitian tugas akhir ini.

2. Bab II – Proses dan Pengembangan Desain

Bab II berisi tinjauan pustaka dari sistem dan subsistem yang dibuat, persyaratan desain yang terdiri dari objektif beserta constraint-nya, konsep desain, dan pemilihan alternatif desain untuk pengembangan desain.

3. Bab III – Detil desain dan Implementasi

Bab III berisi model desain masing-masing bagian dari subsistem yang dibuat, yaitu desain *frontend*, *backend*, dan database, serta implementasinya untuk pembuatan *web dashboard*.

4. Bab IV – Pengujian dan Analisis

Bab IV berisi pengujian *web dashboard*, meliputi pengujian *frontend* (*user interface*) dan pengujian *backend* (kinerja *web dashboard* dan *backup data*).

5. Bab V – Simpulan dan Saran

Bab V berisi simpulan dari penelitian tugas akhir ini dan saran untuk pengembangan lebih lanjut.

BAB II PROSES DAN PENGEMBANGAN DESAIN

2.1 Tinjauan Pustaka

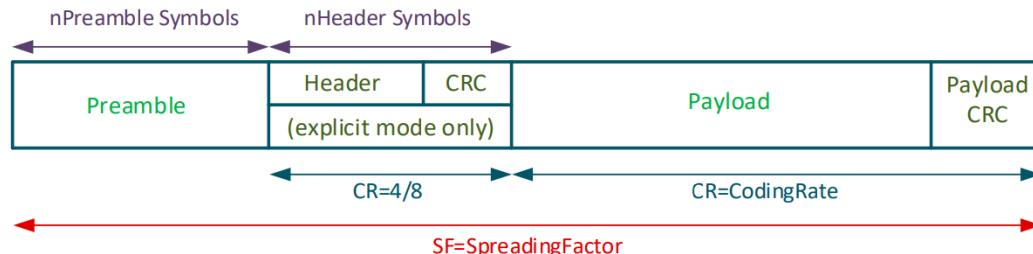
2.1.1 Teknologi LoRa dan LoRaWAN

LoRa (*Long Range*) merupakan teknologi komunikasi nirkabel yang dapat mengirimkan data jarak jauh dengan konsumsi daya rendah pada area yang luas (LPWAN). Teknologi ini sangat cocok digunakan dalam aplikasi IoT yang sering mengirimkan data berukuran kecil, dibandingkan dengan teknologi komunikasi seluler tradisional yang dapat mengirimkan data berukuran besar dengan konsumsi daya tinggi. LoRa juga memiliki jangkauan yang lebih jauh dibandingkan dengan teknologi komunikasi nirkabel konsumsi daya rendah lainnya seperti Wifi, Bluetooth, dan Zigbee. Oleh karena itu, LoRa saat ini menjadi teknologi pendukung utama dalam bisnis IoT yang memerlukan konsumsi daya rendah dan jangkauan yang jauh (Guangyang dkk., 2022).

LoRa dapat dijalankan pada pita frekuensi sub-gigahertz yang bebas lisensi, seperti 915 MHz, 868 MHz, dan 433 MHz. Selain itu, LoRa juga dapat dijalankan pada 2,4 GHz untuk mencapai kecepatan data yang lebih tinggi dibandingkan dengan pita frekuensi sub-gigahertz, meskipun mengorbankan jangkauannya (The Things Network, t.t.-b). Berdasarkan Peraturan Direktur Jenderal Sumber Daya dan Perangkat Pos dan Informatika No. 3 Tahun 2019, pita frekuensi yang dapat digunakan untuk menjalankan LoRa adalah dalam rentang 920-923 MHz.

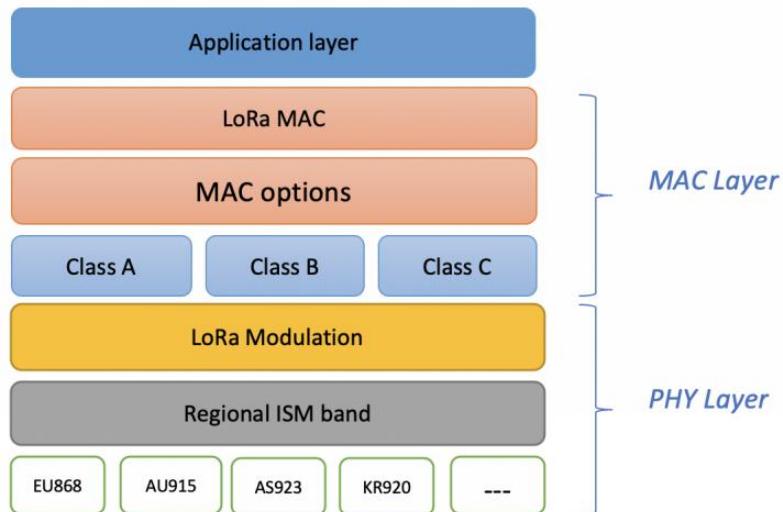
LoRa merupakan teknologi *physical layer* dari LoRaWAN. Dalam komunikasi LoRa, ada beberapa parameter yang dapat diatur, seperti *spreading factor* (SF), *coding rate* (CR) dan *bandwidth* (BW). Nilai dari setiap parameter tergantung pada wilayah di mana perangkat LoRa dipasang. Nilai SF dapat bervariasi antara 7 dan 12. Semakin tinggi nilai SF, semakin tinggi pula nilai rasio *signal-to-noise* (SNR), sensitivitas, jangkauan cakupan, waktu simbol (Ts), dan waktu di udara (ToA / Time of Air) yang menunjukkan durasi pengiriman paket. Modulasi LoRa dapat mengirimkan *frame* secara acak menggunakan dua jenis format paket, yaitu eksplisit dan implisit. Secara umum, struktur paket terdiri dari empat elemen yaitu

preamble, *header* (opsional), *payload* (dibatasi hingga 255 byte) dan *cyclic redundancy check* (CRC) (opsional) (Rivera Guzmán dkk., 2022). Format paket LoRa ditunjukkan pada gambar berikut.



Gambar II.1 Format Paket LoRa

LoRaWAN (*Long Range Wide Area Network*) merupakan protokol *Media Access* (MAC) *layer* yang dibangun di atas teknik modulasi LoRa. Protokol ini membantu *hardware* LoRa dalam pembuatan format pesan informasi dan pengirimannya. Protokol ini juga mengatur komunikasi *end device* LoRa dengan LoRaWAN gateway serta mengelola jaringan, keamanan, dan manajemen data (The Things Network, t.t.-b).



Gambar II.2 Susunan Protokol LoRaWAN

Protokol LoRaWAN memiliki tiga layer utama yaitu *physical layer*, *MAC layer*, dan *application layer*. Pada *physical layer*, modulasi LoRa menggunakan teknik

Chirp Spread Spectrum (CSS) yang dikembangkan oleh Semtech untuk komunikasi jarak jauh (Baddula dkk., 2020). Pada *physical layer* juga terdapat *regional ISM band* yang telah ditetapkan seperti EU868, EU433, CN780 dan AS923. Indonesia masuk dalam *regional ISM band* AS923.

Dalam pengiriman data melalui LoRa, terdapat batasan dalam *bit rate* dan *maximum payload* yang telah diatur berdasarkan parameter LoRa tertentu, berikut tabelnya.

Tabel II.1 Tabel *Bit Rate* dan *Maximum Payload* untuk Parameter LoRa Tertentu

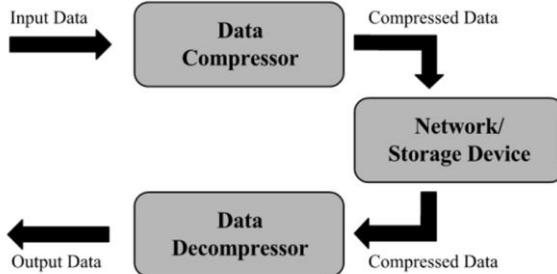
Data rate	Konfigurasi LoRa SF/BW	Bits/s	Maximum payload
DR0	SF12/125kHz	250	59
DR1	SF11/125kHz	440	59
DR2	SF10/125kHz	980	59
DR3	SF9/125kHz	1 760	123
DR4	SF8/125kHz	3 125	230
DR5	SF7/125kHz	5 470	230

Jaringan LoRaWAN diterapkan dengan topologi *stars-of-stars*. *End device* LoRa berkomunikasi dengan dengan LoRaWAN gateway terdekat, dan setiap LoRaWAN gateway terhubung ke server jaringan. Jaringan ini menggunakan protokol berbasis ALOHA, sehingga *end device* LoRa tidak perlu melakukan *peering* dengan LoRaWAN gateway tertentu, karena pesan yang dikirim dari *end device* LoRa akan dikirim ke semua LoRaWAN gateway yang berada dalam jangkauan. Pesan-pesan tersebut akan diterima oleh server dan dapat dideteksi apabila ada salinan pesan yang sama, sehingga pesan salinan tersebut akan dihapus dan menyisakan satu pesan saja pada server (The Things Network, t.t.-a).

2.1.2 Kompresi dan Dekompresi Data

Teknik kompresi data digunakan untuk mengurangi ukuran data yang memungkinkan penyimpanan pada ruang terbatas dan pengiriman melalui jaringan dengan *bandwidth* terbatas (Gupta dkk., 2017). Dekompresi merupakan kebalikan dari kompresi, yaitu mengembalikan data yang telah dikirim atau disimpan menjadi

ukuran semula sehingga didapatkan data aslinya. Berikut merupakan gambaran penggunaan sistem kompresi data secara umum.



Gambar II.3 Komponen Sistem Kompresi Data

Teknik kompresi dapat dibagi menjadi dua jenis, yaitu kompresi *lossless* dan kompresi *lossy*. Teknik kompresi *lossless* tidak menyebabkan hilangnya informasi atau data. Kompresi ini dilakukan dengan merepresentasikan file menggunakan jumlah bit yang lebih sedikit, tanpa menghilangkan informasi. Teknik kompresi *lossy* menghapus bit-bit yang tidak diperlukan pada data, sehingga mengurangi ukuran file dengan cukup signifikan. File yang dikompresi menggunakan teknik ini tidak dapat dikembalikan ke bentuk aslinya karena sebagian informasi hilang saat dalam proses kompresi. Karena banyak informasi yang tidak diperlukan dihapus pada teknik kompresi ini, nilai rasio kompresi ini biasanya lebih tinggi dibandingkan dengan teknik kompresi *lossless* (Gupta dkk., 2017).

Terdapat banyak algoritma kompresi yang tersedia pada saat ini. Diantaranya terdapat algoritma kompresi miniLZO, LZ4, dan Unishox2.

2.1.2.1 *miniLZO*

LZO (Lempel-Ziv-Oberhumer) adalah *library* kompresi data *lossless* portabel yang ditulis dalam ANSI C. Algoritma ini menawarkan kompresi dan dekompresi yang cukup cepat. Pada *library* LZO juga terdapat beberapa level kompresi yang lebih lambat untuk mencapai nilai rasio kompresi yang baik. miniLZO adalah subset yang sangat ringan dari *library* LZO yang dirancang untuk memudahkan

penerapannya dalam berbagai aplikasi. miniLZO juga memiliki fungsi-fungsi penting yang tersedia pada *library* LZO (Oberhumer, 2017).

2.1.2.2 LZ4

LZ4 merupakan algoritma kompresi data *lossless* yang cepat dan efisien. Algoritma ini dikembangkan berdasarkan algoritma LZ77. LZ4 memprioritaskan kecepatan kompresi dan dekompresi dengan mengorbankan nilai rasio kompresi yang lebih rendah dibandingkan beberapa algoritma kompresi lainnya. Algoritma ini sering digunakan dalam aplikasi yang mementingkan kecepatan daripada rasio kompresi, seperti pada pemrosesan data *real-time* dan *file system* (LZ4, t.t.).

2.1.2.3 Unishox2

Unishox2 merupakan algoritma kompresi yang dirancang khusus untuk teks pendek, tidak seperti LZ4, miniLZO, dan algoritma lainnya yang dirancang untuk tujuan umum. Algoritma ini menggabungkan teknik *entropy*, *dictionary*, dan *delta coding* untuk menghasilkan kompresi yang efisien. Unishox2 dapat diaplikasikan untuk pengiriman data teks dengan memori yang rendah pada jaringan kecepatan rendah dan bandwidth terbatas seperti LoRa dan *bluetooth low energy* (siara-cc, t.t.).

2.1.3 Database Management System

Database adalah kumpulan data dan informasi yang disimpan secara terorganisir untuk memudahkan akses, pengelolaan, dan pembaruan data. Data dalam database dapat diambil, ditambahkan, dihapus, atau dimodifikasi dengan melakukan kueri terhadap data tersebut.

Database Management System (DBMS) merupakan *software* yang dapat digunakan untuk membuat dan mengelola database. DBMS dapat menghasilkan laporan, mengendalikan operasi baca dan tulis data, serta melakukan analisis penggunaannya. DBMS berfungsi sebagai *interface* antara *end user* dan *database* untuk memudahkan pengorganisasian dan pembaruan data. Fungsi utama DBMS untuk mengelola data, mengatur struktur logis database (*database schema*), dan mengoperasikan database yang memungkinkan akses terhadap data di dalamnya

(*database engine*). Ketiga fungsi utama tersebut membantu memastikan bahwa pengelolaan data berjalan dengan lancar, mendukung pemrosesan bersamaan (*concurrency*), pemulihan data (*recovery*), keamanan (*security*), dan integritas data (*integrity*) (Hassan, 2021).

Pada awalnya, sistem database tradisional menggunakan model relasional untuk penyimpanan data yang dikenal berdasarkan sebagai database SQL (*Structured Query Language*) karena bahasa yang digunakan untuk mengakses database tersebut (*query*). Namun, seiring berjalannya waktu, mulai berkembang database non-relasional atau yang dikenal sebagai database NoSQL (Li & Manoharan, 2013).

Database relasional sangat terstruktur, menyimpan data dalam tabel-tabel yang terdiri dari baris dan kolom, yang sering disebut sebagai *tuples* dan *attributes*. Tabel-tabel ini saling terhubung melalui *primary key* dan *foreign key*, yang menjaga “*Referential Integrity*” (hubungan konsisten) di antara data, sehingga tabel-tabel ini juga disebut sebagai relasi (Hassan, 2021).

Database relasional memiliki keunggulan dalam fleksibilitas, kesederhanaan, akses *multi-user*, dan adanya dukungan bahasa standar (SQL), sehingga tetap banyak digunakan hingga saat ini. Namun, database relasional juga memiliki beberapa kelemahan, termasuk keterbatasan dalam skalabilitas untuk menangani pertumbuhan data yang eksponensial, biaya tinggi dalam pengaturan dan pemeliharaan, serta kesulitan dalam menangani data tak terstruktur dan berbagi informasi antar database (Hassan, 2021).

Dengan meningkatnya aksesibilitas internet dan ketersediaan penyimpanan data yang lebih murah, berbagai jenis data—terstruktur, semi-terstruktur, dan tak terstruktur—dapat lebih mudah dikumpulkan dan disimpan untuk berbagai keperluan. Data-data ini sering disebut *Big Data*. Untuk mengelola jumlah data yang sangat besar, diperlukan sistem yang cepat, skema yang fleksibel, dan database yang terdistribusi (tidak terpusat). Database NoSQL muncul sebagai solusi

utama untuk *Big Data* karena mampu memenuhi kebutuhan ini, yang menyebabkan banyaknya pilihan database NoSQL yang tersedia (Li & Manoharan, 2013).

2.1.3 Dashboard Berbasis Web

Dashboard adalah alat untuk merepresentasikan data dalam bentuk visual secara komprehensif pada satu halaman untuk memudahkan kontrol informasi secara cepat. Dashboard yang baik dapat membantu melihat gambaran besar dari data. Fitur penting yang harus ada dalam dashboard yaitu menampilkan data penting untuk dipantau, mudah digunakan oleh semua orang, dan data harus dapat diperbaiki secara otomatis (Hidayati dkk., 2022).

Pada media digital, dashboard dapat dibuat dalam beberapa cara, seperti melalui Aplikasi *mobile* (*smartphone*), aplikasi *desktop* (PC atau laptop), dan aplikasi web. Aplikasi *mobile* memiliki kelebihan untuk pemantauan data kapanpun dan dimanapun, namun memiliki keterbatasan dalam hal tampilan yang sempit dan interaksi yang terbatas. Aplikasi *desktop* menawarkan performa yang sangat baik dan fitur yang lebih kompleks, namun pembuatan dan penggunaannya cenderung lebih susah, karena kurang portabel dan memerlukan instalasi yang rumit. Aplikasi web memiliki kelebihan untuk digunakan pada perangkat apapun, baik pada perangkat *mobile* maupun *desktop*. Aplikasi ini juga dapat menawarkan performa yang lebih baik dengan fitur yang beragam, namun kinerjanya sangat bergantung pada web browser dan web server yang digunakan.

Aplikasi web terdiri dari dua bagian utama, yaitu browser *frontend* dan server *backend* yang keduanya tersebut terhubung melalui sebuah jaringan. Browser *frontend* menampilkan konten pada halaman web dan menerima permintaan dari user, lalu mengirimkan permintaan tersebut ke server *backend*. Server *backend* kemudian memproses permintaan tersebut dan mengirimkan hasilnya kembali ke browser. Proses ini disebut “*request response*” (He, 2023).

Frontend berkomunikasi melalui jaringan dengan protokol HTTP/HTTPS untuk mengirimkan *request* ke *backend* dan menerima respons. API (*Application Programming Interface*) merupakan *interface* yang didefinisikan di *backend* untuk

memungkinkan *frontend* mengakses fungsionalitas dan data yang disediakan oleh server. Oleh karena itu, aplikasi web dapat lebih interaktif dengan pengguna.

Untuk membuat dan mengembangkan *frontend*, digunakan berbagai alat dan teknologi yang biasanya melibatkan HTML, CSS, dan JavaScript. Ketiga teknologi ini bekerja sama di dalam browser untuk mengendalikan tampilan dan interaksi pada halaman web. Sementara itu, *backend* biasanya melibatkan tiga komponen utama, yaitu server, aplikasi, dan database. Teknologi yang digunakan pada *backend* biasanya menggunakan bahasa pemrograman seperti PHP, Ruby, Python, dan lainnya (Abdullah & Zeki, 2014).

Dalam pembuatan *frontend*, *framework* yang populer digunakan yaitu React, Angular, Vue.js, dan Bootstrap. Sedangkan untuk *backend*, *framework* yang populer digunakan yaitu Node.js, Django, Flask, Ruby on Rails, dan Laravel.

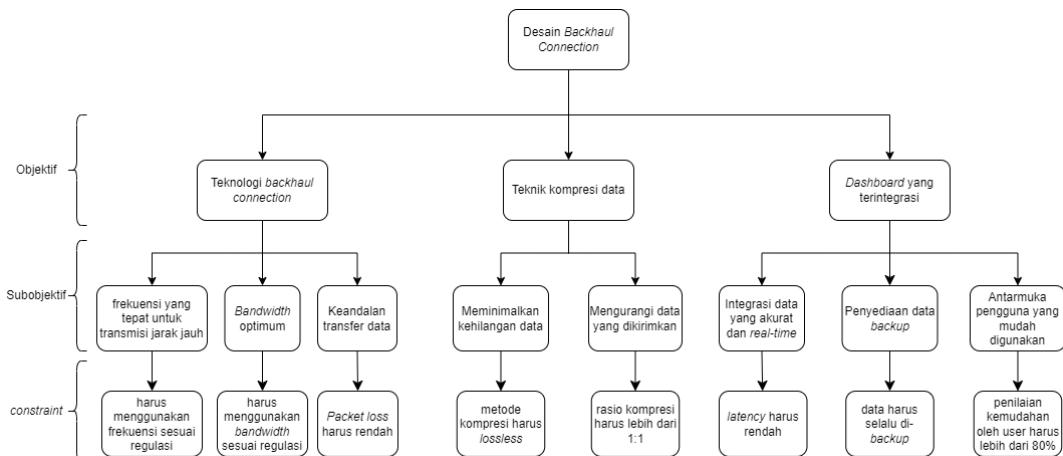
2.2 Persyaratan Desain

Pada tugas akhir ini, pertama-tama ditentukan objektif-objektif yang harus dipenuhi ketika mendesain sistem berdasarkan kebutuhan. Objektif-objektif yang ditentukan yaitu sebagai berikut:

1. Teknologi *backhaul connection* yang tepat untuk daerah suburban atau rural
2. Teknik kompresi data yang efisien
3. *Dashboard* yang terintegrasi

Objektif-objektif yang telah didefinisikan tersebut dirasa masih bersifat umum. Oleh karena itu, dilakukan penjabaran setiap objektif dalam bentuk subobjektif. Objektif pertama yaitu teknologi *backhaul connection* yang tepat, dijabarkan menjadi 3 subobjektif yaitu keandalan transfer data, penggunaan bandwidth yang optimum, dan penggunaan frekuensi kerja yang sesuai dengan regulasi. Objektif kedua yaitu teknik kompresi data yang efisien, dijabarkan menjadi 2 subobjektif yaitu meminimalkan kehilangan data selama proses kompresi dan dekompresi dan mengurangi ukuran data yang dikirimkan. Objektif ketiga yaitu *dashboard* yang terintegrasi, dijabarkan menjadi tiga subobjektif yaitu integrasi data yang akurat dan *real-time*, penyediaan data *backup* yang telah didekompresi, dan antarmuka pengguna yang mudah digunakan.

Selanjutnya, dari tiap subobjektif yang telah dijelaskan sebelumnya, didefinisikan sebuah *constraint* agar mendapatkan kejelasan terkait parameter keberhasilan yang akan diturunkan selanjutnya. Subobjektif pertama yaitu penggunaan frekuensi yang tepat untuk transmisi jarak jauh, *constraint* yang didefinisikan yaitu harus menggunakan frekuensi sesuai regulasi pemerintah Indonesia. Subobjektif selanjutnya yaitu penggunaan *bandwidth* yang optimum, *constraint* yang didefinisikan yaitu harus menggunakan *bandwidth* sesuai dengan regulasi pemerintah. Subobjektif selanjutnya yaitu keandalan transfer data, *constraint* yang didefinisikan yaitu *packet loss* harus rendah. Subobjektif selanjutnya yaitu meminimalkan kehilangan data, *constraint* yang didefinisikan yaitu metode kompresi harus *lossless*. Subobjektif selanjutnya yaitu mengurangi data yang dikirimkan, *constraint* yang didefinisikan yaitu rasio kompresi harus lebih dari 1:1. Subobjektif selanjutnya yaitu integrasi data yang akurat dan *real-time*, *constraint* yang didefinisikan yaitu *latency* harus rendah. Subobjektif selanjutnya yaitu penyediaan data *backup*, *constraint* yang didefinisikan yaitu data harus selalu di-*backup*. Subobjektif selanjutnya yaitu antarmuka pengguna yang mudah digunakan, *constraint* yang didefinisikan yaitu penilaian kemudahan oleh *user* harus lebih dari 80%. Penjelasan singkat mengenai objektif, subobjektif, dan *constraint* dapat dilihat pada gambar berikut.



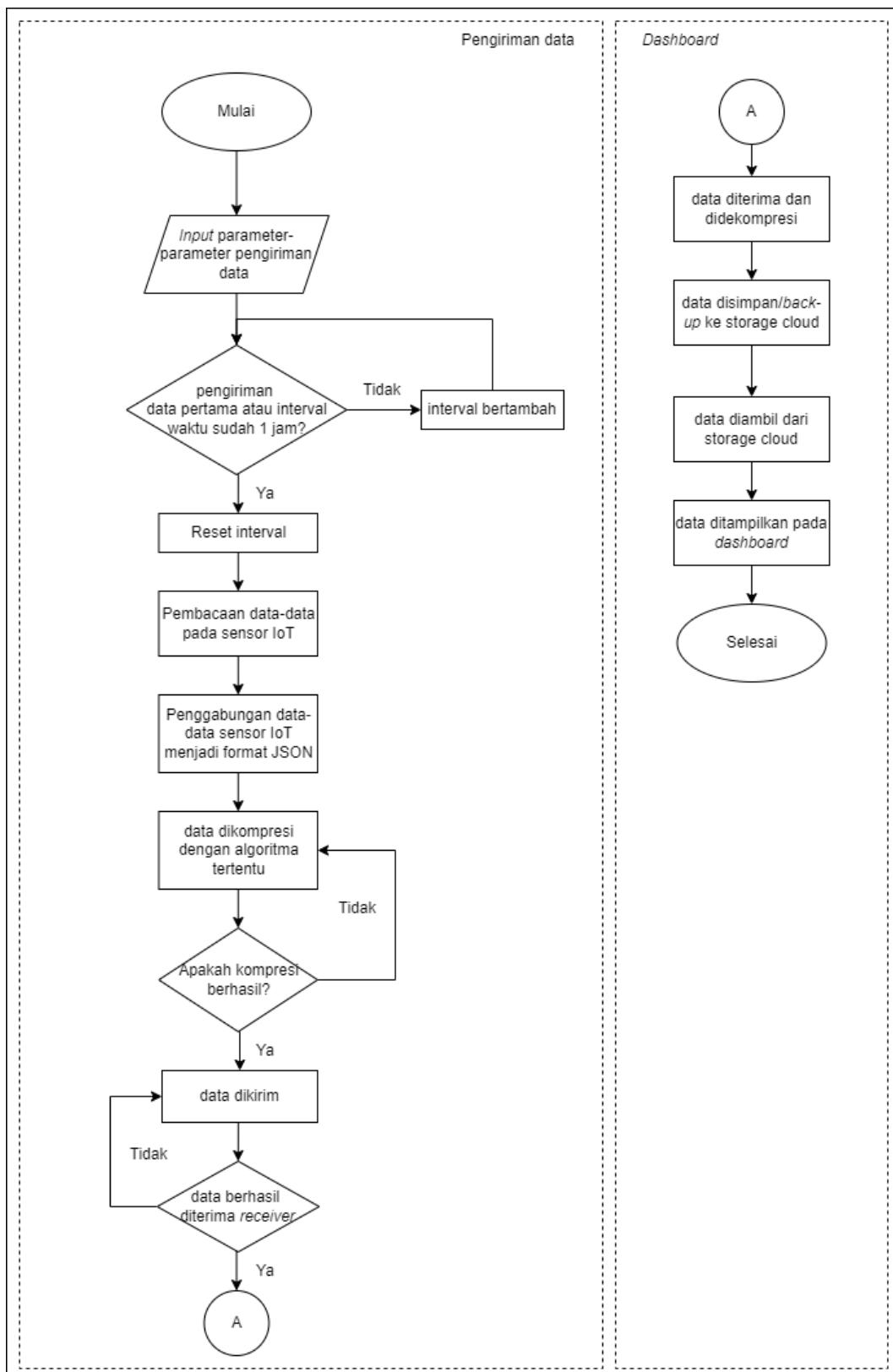
Gambar II.4 Pohon Objektif

Subobjektif dan constraint yang telah ditentukan kemudian digunakan untuk merumuskan persyaratan fungsional yang bertujuan untuk mengelaborasikan subobjektif dalam bentuk fungsi. Selanjutnya, berdasarkan fungsi-fungsi tersebut, akan ditentukan beberapa subsistem yang merepresentasikan fungsi-fungsi tersebut seperti yang dijabarkan pada tabel berikut.

Tabel II.2 Tabel Persyaratan Fungsional dan Fungsi

Subobjektif & Constraint	Persyaratan Fungsional	Fungsi
<p>Subobjektif 1 : Penggunaan frekuensi yang tepat untuk transfer data jarak jauh.</p> <p>Constraint : Harus menggunakan frekuensi yang dapat digunakan di Indonesia</p> <p>Subobjektif 2 : <i>Bandwidth</i> optimum.</p> <p>Constraint : Tidak boleh lebih besar dari <i>bandwidth</i> maksimum yang diizinkan oleh regulasi di Indonesia</p> <p>Subobjektif 3 : Kendalan dalam pengiriman data</p> <p>Constraint : <i>packet loss</i> harus rendah</p> <p>Subobjektif 4 : Meminimalkan kehilangan data</p> <p>Constraint : metode kompresi harus <i>lossless</i></p> <p>Subobjektif 5 : Mengurangi data yang dikirimkan</p>	<ol style="list-style-type: none"> 1. Menggunakan frekuensi untuk transfer data jarak jauh sesuai regulasi Menteri Komunikasi dan Informatika. 2. Mengirimkan data perangkat IoT dengan antena yang memiliki frekuensi kerja tertentu dan memiliki bandwidth yang optimum. 3. Mengirimkan data perangkat IoT dari kolam ke e-Fishery point dengan <i>packet loss</i> yang rendah. 4. Mengimplementasikan algoritma kompresi <i>lossless</i> untuk memastikan tidak ada data yang hilang selama proses kompresi dan dekompresi. 5. Menggunakan algoritma kompresi yang menghasilkan rasio kompresi lebih besar dari 1:1 untuk mengurangi ukuran data tanpa mengurangi kualitas dan akurasi data. 	<p>Pengiriman data: 1,2,3,4,5</p> <p>Dashboard: 6,7,8</p>

Subobjektif & Constraint	Persyaratan Fungsional	Fungsi
<p>Constraint : rasio kompresi harus lebih dari 1:1</p> <p>Subobjektif 6 : Integrasi data yang akurat dan <i>real-time</i></p> <p>Constraint : <i>latency</i> harus rendah</p> <p>Subobjektif 7 : Penyediaan data <i>backup</i></p> <p>Constraint : data harus selalu di <i>backup</i></p> <p>Subobjektif 8: Antarmuka pengguna yang mudah digunakan</p> <p>Constraint : penilaian kemudahan oleh <i>user</i> harus lebih dari 80%.</p>	<p>6. Menggunakan protokol komunikasi <i>real-time</i> untuk memastikan data dari sensor diterima dan ditampilkan di dashboard dengan <i>latency</i> rendah, idealnya di bawah 1 detik.</p> <p>7. Mengimplementasikan mekanisme <i>backup</i> otomatis setiap interval waktu tertentu (misalnya setiap hari) untuk memastikan data sensor IoT yang telah didekompresi selalu tersedia dan tidak hilang.</p> <p>8. melakukan uji coba dengan pengguna untuk memastikan bahwa tingkat kepuasan pengguna terhadap kemudahan penggunaan antarmuka mencapai lebih dari 80%.</p>	



Gambar II.5 Flowchart Sistem Terintegrasi

Gambar di atas menunjukkan tingkah laku sistem yang terdiri dari dua buah subsistem, yaitu pengiriman data dan *dashboard*. Dari diagram alir tingkah laku tersebut, terlihat hubungan antara dua buah subsistem yang digunakan. Secara umum, subsistem pengiriman data akan mengirim data yang telah dikompresi ke server tujuan. Sementara subsistem *dashboard* akan mem-backup data yang telah didekompresi ke *cloud storage* dan menampilkannya dalam bentuk *dashboard*.

2.3 Konsep Desain

Pada konsep desain, ditentukan metrik pengukuran dan syarat pemenuhan berdasarkan subobjektif yang didefinisikan sebelumnya, yang diambil dari sebuah standar, buku, atau publikasi tertentu, seperti yang tertera pada tabel berikut.

Tabel II.3 Tabel Metrik Pengukuran dan Syarat Pemenuhan

Subobjektif	Metrik Pengukuran	Syarat Pemenuhan
Penggunaan frekuensi yang tepat untuk transfer data jarak jauh.	Nilai eksak frekuensi kerja perangkat yang digunakan dalam transmisi.	Berdasarkan PERDIRJEN SDPPI NO 3 TAHUN 2019 LPWA Menkominfo (Direktur Jendral Sumber Daya dan Perangkat Pos dan Informatika, 2019). Pita Frekuensi Radio untuk perangkat LPWA Nonseluler : 920 - 923 MHz
<i>Bandwidth</i> optimum.	Nilai eksak bandwidth yang dihasilkan dan digunakan.	Berdasarkan PERDIRJEN SDPPI NO 3 TAHUN 2019 LPWA Menkominfo (Direktur Jendral Sumber Daya dan Perangkat Pos dan Informatika, 2019). Bandwidth LPWA Nonseluler Wideband : \leq 250 kHz Narrowband : \leq 200 kHz
Kendalan dalam pengiriman data	Nilai persentase <i>packet loss</i> .	Standar <i>packet loss</i> (Sukmandhani, 2020). Sangat bagus : 0% Bagus : 3% Sedang : 15% Jelek : 25%

Subobjektif	Metrik Pengukuran	Syarat Pemenuhan
Meminimalkan kehilangan data	Keberhasilan proses kompresi dan dekompresi	Berdasarkan standar dari jurnal "A Survey of Lossless Compression Techniques" yang menyebutkan bahwa metode kompresi lossless harus memiliki persentase kehilangan data 0% (Rani & Singh, 2016).
Mengurangi data yang dikirimkan	Rasio kompresi data	Rasio kompresi harus lebih dari 1:1 (siara-cc, t.t.).
Integrasi data yang akurat dan <i>real-time</i>	<i>Latency</i> saat transmisi data	<i>Latency</i> harus <50ms berdasarkan standar ITU-T G.114
Penyediaan data <i>backup</i>	Frekuensi <i>backup</i> data	Data harus di- <i>backup</i> secara harian sesuai dengan praktik terbaik industry (Swanson dkk., 2010).
Antarmuka pengguna yang mudah digunakan	Penilaian kemudahan oleh pengguna	Antarmuka yang ramah pengguna sesuai dengan standar kegunaan yaitu ISO 9241.

2.4 Pengembangan Desain

Dengan terbaginya penelitian menjadi dua subsistem, diperlukan riset untuk pemilihan teknologi dan alat yang sesuai dalam mengembangkan dan mengintegrasikan subsistem yang akan dibuat. Pada subsisten *dashboard*, dilakukan riset terkait teknologi dan alat untuk pembuatan aplikasi web yang terdiri dari bagian *frontend* dan *backend*. Pemilihan teknologi dan alat tersebut diharapkan dapat digunakan untuk memenuhi persyaratan fungsional yang telah dijelaskan sebelumnya.

2.4.1 Alternatif Desain Frontend

Frontend dari aplikasi web bertanggung jawab menampilkan halaman web dan berinteraksi dengan pengguna. Dalam pengembangan *frontend*, beberapa *framework* populer yang sering dipertimbangkan yaitu React, Angular, dan

Bootstrap. Masing-masing *framework* tersebut memiliki kelebihan dan kekurangan sebagai berikut.

React merupakan library JavaScript yang dikembangkan oleh Facebook untuk membangun *user interface*. React memiliki kelebihan yaitu dapat mengelola *state* dengan efisien, virtual DOM yang dapat meningkatkan performa aplikasi, dan ekosistem yang kaya dengan berbagai *plugin* dan *tools*. Namun, React memerlukan pembelajaran tambahan mengenai konsep *state management* dan JSX, sehingga memiliki kurva pembelajaran yang cukup curam.

Angular merupakan *framework* JavaScript yang dikembangkan oleh Google. Angular menawarkan solusi *all-in-one* untuk membangun aplikasi web, termasuk pengelolaan *routing*, formulir, dan *state management*. Kelebihan dari *framework* ini adalah dibangun di atas TypeScript untuk pengelolaan kode yang lebih baik serta dukungan bawaan untuk banyak fitur yang diperlukan dalam pengembangan aplikasi, sehingga sangat cocok untuk proyek skala besar dan aplikasi *enterprise*. Namun, Angular lebih kompleks dari *framework* lainnya, yang mengakibatkan kurva pembelajarannya yang lebih curam. Selain itu, *framework* ini membutuhkan lebih banyak *resource* daripada *framework* lainnya.

Bootstrap merupakan *framework* CSS yang digunakan untuk membangun aplikasi web yang responsif dan mudah digunakan. Bootstrap menyediakan berbagai komponen *user interface* dan *style* yang dapat mempercepat pengembangan *frontend*. Bootstrap juga memiliki kurva pembelajaran yang relatif landai karena mudah dipahami. Namun, Bootstrap lebih fokus pada *styling* dan desain daripada pada interaktivitas atau *state management*.

Menggunakan HTML, CSS, dan JavaScript standar memiliki kelebihan dalam fleksibilitas dan kontrol penuh terhadap struktur, *style*, dan perilaku dalam pembuatan tampilan aplikasi web. Kode yang dihasilkan juga cenderung lebih ringan karena tidak ada overhead dari *library* tambahan atau fitur yang tidak diperlukan, sehingga menghasilkan kinerja yang lebih baik bagi aplikasi web.

Kekurangannya yaitu dalam manajemen kode yang lebih rumit dan lamanya pengembangan pada proyek besar jika tanpa bantuan *framework*.

2.4.2 Alternatif Desain Backend

Backend bertanggung jawab dalam mengatur logika aplikasi, manajemen data, dan komunikasi dengan database. Beberapa *framework* populer yang sering dipertimbangkan untuk pengembangan *backend* adalah Express.js, Django, Flask, dan Laravel. Masing-masing *framework* tersebut memiliki keunggulan dan kelemahan sebagai berikut.

Express.js merupakan *framework backend* populer untuk ekosistem Node.js. Keunggulannya yaitu dapat membangun aplikasi web dengan cepat dengan bantuan *middleware* yang terintegrasi di dalamnya, serta dukungan yang kuat dari komunitasnya. Express.js cocok untuk aplikasi-aplikasi yang membutuhkan skalabilitas dan performa tinggi. Kekurangan utama dari Express.js yaitu tidak menyediakan struktur atau konvensi yang ketat untuk pengaturan kode.

Django merupakan *framework* berbasis Python yang menggunakan pola arsitektur *model-view-controller*. *Framework* ini berfokus untuk pengembangan aplikasi web kompleks berbasis database. Kelebihan utamanya adalah ORM (*Object-Relational Mapping*) yang kuat dan integrasi yang baik dengan fitur bawaan seperti autentikasi pengguna, admin panel, dan keamanan. Django adalah solusi yang baik untuk aplikasi web dengan performa tinggi dalam proyek yang besar. Kekurangan Django yaitu memiliki kurva pembelajaran yang cukup curam karena cukup kompleks dengan banyaknya fitur yang dimilikinya.

Flask adalah *framework* desain web yang ringan dan fleksibel berbasis Python. Flask dirancang sebagai *micro-framework*, yang memberikan kebebasan kepada pengembang web untuk memilih komponen dan *tools* yang ingin digunakan. Meskipun Flask tidak memiliki fitur bawaan sebanyak Django, Flask dapat menambahkan fitur tambahan yang dibutuhkan, sehingga memungkinkan penyesuaian yang lebih baik dengan kebutuhan spesifik aplikasi.

Laravel adalah *framework* PHP yang menawarkan berbagai fitur modern dan kemudahan penggunaan seperti Eloquent ORM dan sistem *routing* yang intuitif. Laravel cocok untuk aplikasi PHP dengan kebutuhan pengembangan yang lebih struktural dan fitur-fitur canggih. Namun, penggunaannya bergantung pada kebutuhan dan preferensi bahasa pemrograman yang digunakan.

2.4.3 Alternatif Database

Dalam pengembangan aplikasi web, database berperan penting dalam menyimpan, mengelola, dan mengakses data yang digunakan oleh aplikasi. Database memastikan data tetap konsisten dan valid dengan menyediakan berbagai fitur seperti *constraint* dan *referential integrity*, yang menjaga keamanan dan integritas data. Secara struktur, terdapat dua jenis database, yaitu database relasional (SQL) dan database non-relasional (NoSQL).

Database SQL menggunakan skema terstruktur dengan tabel dan relasi, menawarkan keuntungan seperti konsistensi data melalui transaksi ACID (*Atomicity, Consistency, Isolation, Durability*), struktur data yang jelas, kemampuan menjalankan query kompleks dengan SQL, serta integritas data yang dijaga melalui *constraint*. Selain itu, database SQL memiliki ekosistem yang matang dan dukungan dari komunitas yang besar. Namun, kekurangan dari database SQL yaitu keterbatasan dalam skalabilitas, performa yang menurun saat menangani volume data yang sangat besar, dan beberapa sistem manajemen basis data relasional (RDBMS) komersial memiliki biaya lisensi yang tinggi.

Database NoSQL dirancang untuk fleksibilitas dan skalabilitas horizontal, memungkinkan penanganan volume data yang sangat besar dengan biaya yang lebih rendah. Database NoSQL menawarkan fleksibilitas skema yang memungkinkan penyimpanan data yang tidak terstruktur dan semi-terstruktur, serta kinerja yang tinggi dengan latensi rendah untuk operasi baca-tulis. Database NoSQL juga menyediakan berbagai model data seperti dokumen, *key-value*, dan graf, yang dapat disesuaikan dengan kebutuhan spesifik aplikasi. Namun, kekurangan database NoSQL yaitu konsistensi data yang terbatas, kurangnya standar query seperti SQL, dan fitur integritas data bawaan seperti constraint juga

menjadi sebuah tantangan. Selain itu, dukungan untuk transaksi juga lebih terbatas dibandingkan dengan database relasional, sehingga memerlukan pengelolaan tambahan dari aplikasi.

2.4.4 Pemilihan Desain

Dalam desain *frontend* aplikasi web, penulis memilih alternatif menggunakan HTML, CSS, dan JavaScript standar dengan pemanfaatan *framework* Bootstrap untuk membangun web yang responsif dan mudah digunakan. Pemilihan alternatif tersebut didasarkan pada pemahaman dasar tentang HTML, CSS, dan JavaScript yang juga perlu dipelajari terlebih dahulu jika ingin menggunakan *framework* React, Angular, dan *framework* lain. Selain itu, pembuatan web menggunakan HTML, CSS, dan JavaScript dengan bantuan *framework* Bootstrap akan memiliki kurva pembelajaran yang lebih landai dibandingkan *framework* lain.

Dalam desain *backend* aplikasi web, penulis memilih alternatif *framework* Flask. Pemilihan alternatif tersebut didasarkan pada kebutuhan desain web yang ringan dengan fleksibilitas dalam memilih fitur yang ingin digunakan tanpa harus menginstall banyak fitur bawaan. Selain itu, *framework* ini menggunakan bahasa pemrograman Python yang sudah dipelajari sebelumnya, sehingga penggunaan *framework* lebih mudah dipahami dan memiliki kurva pembelajaran yang landai.

Data sensor IoT memiliki ukuran yang kecil dengan berbentuk teks. Sehingga tidak dibutuhkan penyimpanan dalam bentuk gambar, dokumen, atau objek lainnya. Database relasional (SQL) menjadi pilihan penulis karena lebih terstruktur dan mudah dimodifikasi dengan penggunaan query yang sudah tersedia. Selain itu, integritas data juga dapat terjamin dengan adanya *primary key*, *foreign key*, dan *constraints*.

BAB III DETIL DESAIN DAN IMPLEMENTASI

3.1 Model Desain

Secara garis besar, topik tugas akhir ini akan dibagi menjadi dua buah subsistem yaitu pengiriman data dan *dashboard*. Penjelasan lebih lanjut mengenai rincian desain subsistem dan pembagian kerja dijabarkan pada tabel berikut.

Tabel III.1 Pembagian Subsistem

Subsistem	Rincian Subsistem	Pembagian Kerja
Pengiriman data	Pengiriman data dilakukan dengan LoRa. Data yang dikirim ke LoRaWAN <i>gateway</i> telah dikompresi dengan algoritma tertentu. Data yang terkompres akan tersimpan pada server Antares.	Helmi Ahmad Khaderani
<i>Dashboard</i>	Data akan diambil dari server Antares dan didekompresi. Data yang telah didekompresi akan di- <i>backup</i> pada <i>cloud storage</i> . Lalu data akan ditampilkan pada <i>dashboard</i> .	Muhammad Shafly Nurrasyid

Berdasarkan Tabel III.1, penulis bertanggung jawab untuk mengerjakan subsistem *dashboard*. Subsistem *dashboard* dapat dibagi menjadi tiga bagian, yaitu:

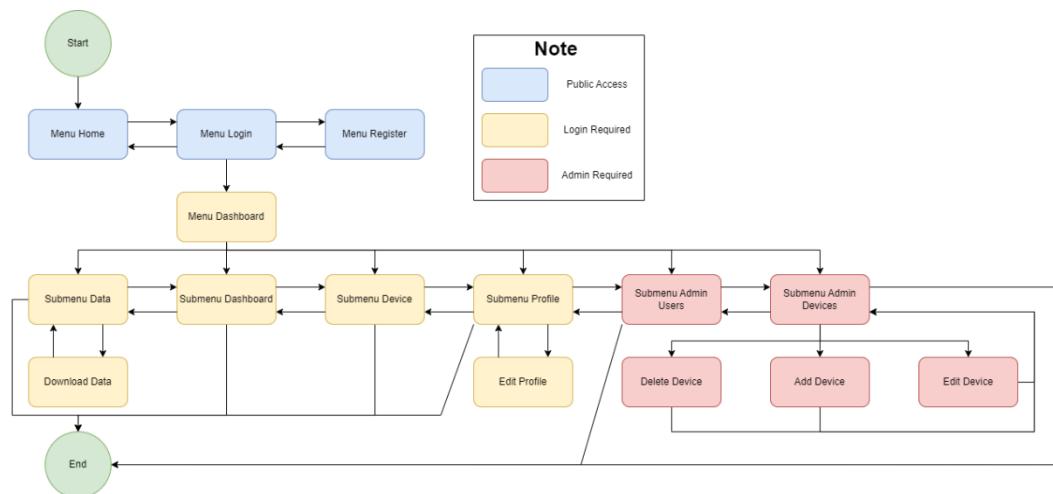
1. Desain *Frontend*
2. Desain *Backend*
3. Desain *Database*

3.1.1 Desain Frontend



Gambar III.1 Arsitektur Frontend

Dalam pengembangan *frontend*, HTML (*Hyper Text Markup Language*) digunakan untuk membuat struktur web, yang mencakup berbagai halaman web berdasarkan menu dan submenu, serta konten di dalamnya yang isinya direspon dari *backend*. Untuk menampilkan halaman web yang lebih menarik secara visual, dilakukan *styling* untuk mengatur teks, *margin*, *background*, *border*, dan animasi menggunakan CSS (*Cascading Style Sheets*). *Styling* dilakukan menggunakan *framework* Bootstrap yang dimodifikasi oleh penulis agar sesuai dengan kebutuhan desain. Selain itu, JavaScript digunakan untuk menambahkan interaktivitas pada halaman web, sehingga membuatnya lebih dinamis dan responsif.



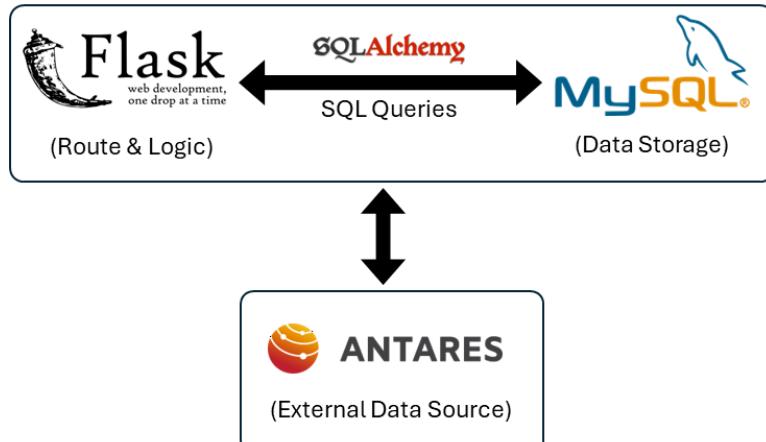
Gambar III.2 Flowchart Navigasi Web Dashboard

User dapat mengakses aplikasi web melalui web browser yang terhubung ke web server. Pertama-tama akan ditampilkan menu Home. Dari menu tersebut, user dapat mengakses menu lain yaitu Login dan Register. Untuk mengakses menu Dashboard, user harus melakukan login terlebih dahulu pada menu Login. Jika belum mempunyai akun, user dapat melakukan registrasi akun melalui menu Register. Pada menu Dashboard, terdapat beberapa submenu yaitu Data, Dashboard, Device, dan Profile. Khusus untuk admin, terdapat tambahan submenu Admin Users dan Admin Devices.

Terdapat beberapa fitur yang tersedia pada submenu tertentu. Pada submenu Data, user dapat mengunduh data dari server database ke penyimpanan lokal. Pada

submenu profile, user dapat mengedit informasi user seperti username, email, dan password. Pada submenu Admin Devices, admin dapat menambahkan, mengedit, dan menghapus device tertentu yang dimiliki user.

3.1.2 Desain Backend



Gambar III.3 Arsitektur Backend

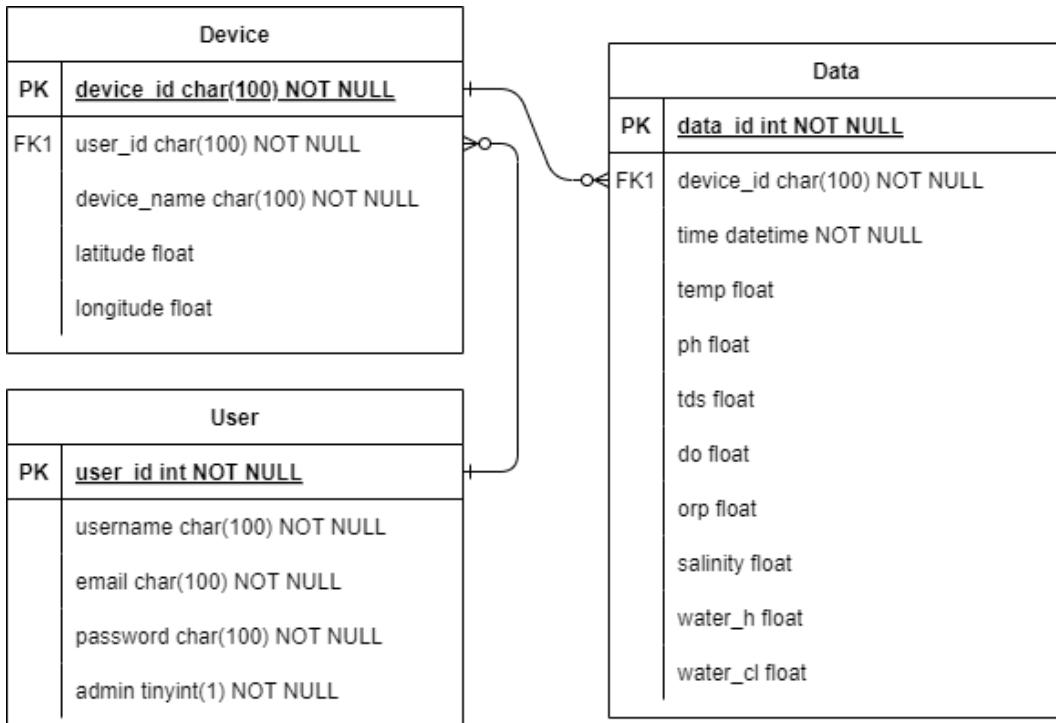
Pada pengembangan *backend*, *framework* Flask digunakan untuk menangani seluruh logika aplikasi web. Flask menyediakan API untuk menangani permintaan dari *frontend*. Flask mengelola berbagai *endpoint* API yang mengatur bagaimana data diambil dari dan dikirim ke database. Untuk menyimpan dan mengelola data, digunakan MySQL sebagai sistem manajemen basis data relasional (RDBMS). Komunikasi antara Flask dan MySQL dilakukan melalui *SQL queries* menggunakan SQLAlchemy.

Untuk data sensor IoT yang tersimpan pada server eksternal Antares, Flask mengintegrasikan *backend* dengan API Antares untuk mengambil data tersebut. Data dari Antares masih dalam bentuk hasil kompresi, sehingga perlu didekompresi terlebih dahulu menggunakan *library* dekompresi Unishox2 sebelum disimpan di MySQL. Setelah disimpan, data ini kemudian diproses dan diatur sesuai kebutuhan sebelum dikirimkan ke *frontend*.

3.1.3 Desain Database

Untuk memastikan desain database yang efisien dan terstruktur dengan baik, dibuat *Entity Relationship Diagram* (ERD). ERD ini membantu dalam memvisualisasikan

struktur database dan hubungan antara tabel-tabel yang ada. Berikut ERD yang telah penulis buat.



Gambar III.4 Entity Relationship Diagram

Tabel User memiliki relasi *one-to-many* dengan tabel Device, artinya setiap pengguna dapat memiliki banyak device, tetapi setiap device hanya terkait dengan satu pengguna. Tabel Device memiliki relasi *one-to-many* dengan tabel Data, yang berarti setiap device dapat menghasilkan banyak entri data, tetapi setiap entri data hanya terkait dengan satu device. Relasi tersebut diimplementasikan melalui *foreign key* dan *relationship* yang ditentukan menggunakan *library SQLAlchemy*. Tabel User dihubungkan dengan tabel Device melalui user_id, sedangkan tabel Device dihubungkan dengan tabel Data melalui device_id.

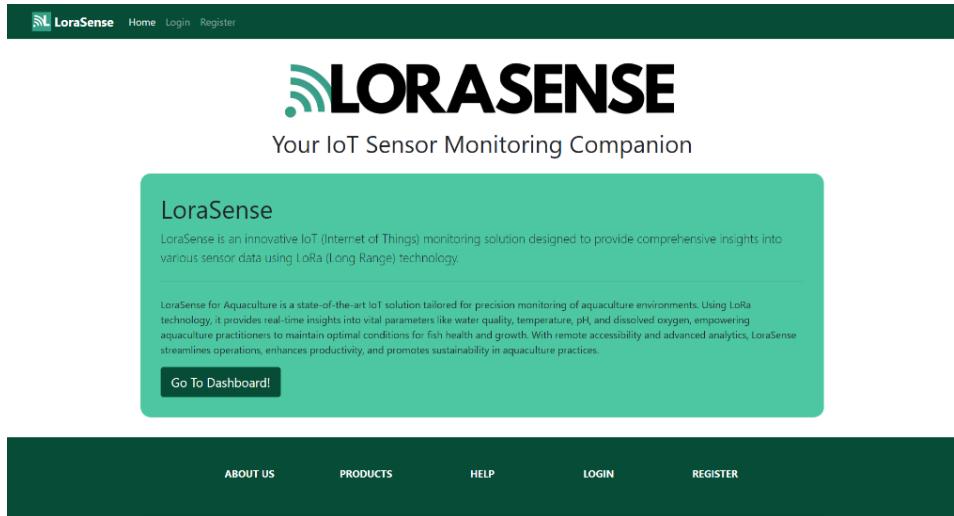
3.2 Implementasi

3.2.1 Implementasi Frontend

Dalam implementasi *frontend*, penulis mengikuti desain yang telah dijelaskan sebelumnya. Pertama-tama, penulis membuat file HTML untuk halaman web dari setiap menu dan submenu yang diperlukan. Lalu, dilakukan *styling* pada halaman

web menggunakan CSS dengan *framework* Bootstrap. Untuk menambahkan interaktivitas dengan user, digunakan JavaScript dengan *library* jQuery dan Chart.js. Berikut ini adalah tampilan halaman web yang telah dibuat.

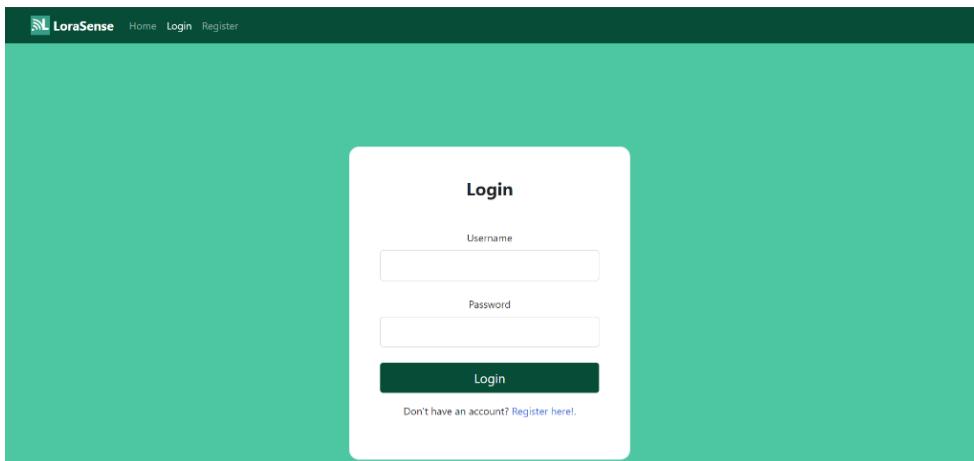
1. Menu Home



Gambar III.5 Menu Home

Menu Home menampilkan informasi umum tentang *web dashboard* yang telah dibuat. yang merupakan dashboard untuk pemantauan data IoT. Pada menu Home, terdapat navbar untuk mengakses menu lain yaitu menu Login dan menu Register. Jika user menekan tombol “Go To Dashboard!” dan user sudah melakukan login, user akan beralih ke menu Dashboard. Namun, jika user belum melakukan login, user akan beralih ke menu Login.

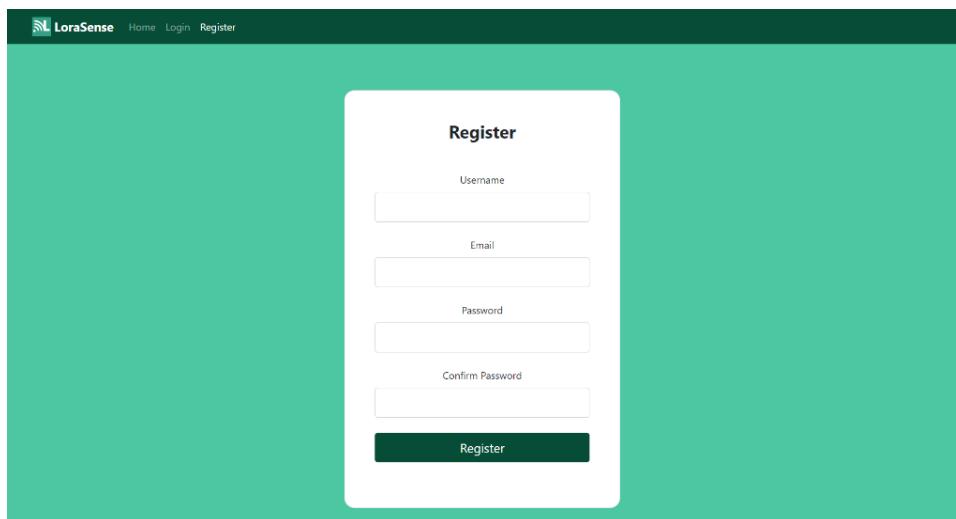
2. Menu Login



Gambar III.6 Menu Login

Menu Login memungkinkan user mengakses akun dengan memasukkan kredensial yang tepat. Menu ini menampilkan formulir login yang meminta user untuk memasukkan username dan password. Jika user berhasil login, user akan diarahkan ke menu Dashboard yang berisi submenu Device, submenu Dashboard, submenu Data, dan submenu Profile. Namun, jika user belum memiliki akun atau tidak memasukkan kredensial yang benar, user akan tetap berada di menu Login.

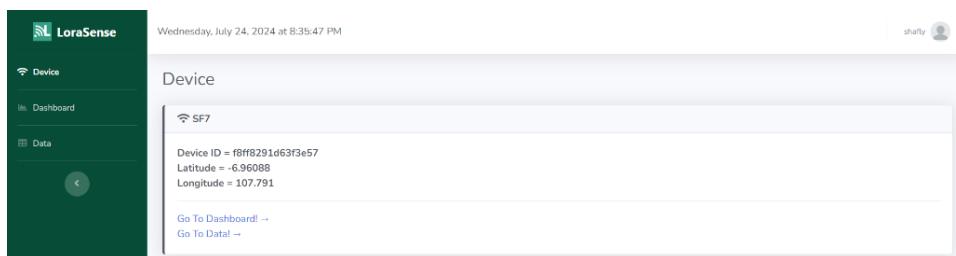
3. Menu Register



Gambar III.7 Menu Register

Menu Register memungkinkan user baru untuk membuat akun. Menu ini menyediakan formulir registrasi yang meminta user mengisi informasi username, email, password, dan konfirmasi password. Jika registrasi berhasil, user akan dibuatkan akun dan diarahkan ke menu Login. Namun, jika terdapat kesalahan dalam proses registrasi atau jika user telah memiliki akun, user akan diberikan pesan kesalahan dan kembali ke menu Register.

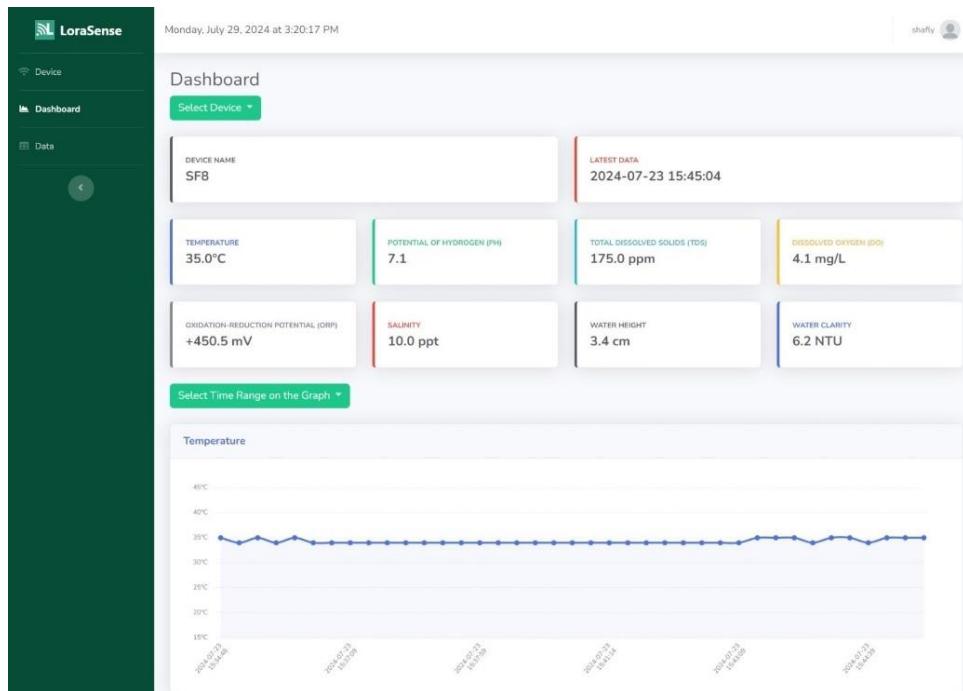
4. Submenu Device



Gambar III.8 Submenu Device

Setelah berhasil login, user akan diarahkan ke menu Dashboard. Menu Dashboard terdiri dari beberapa submenu. Pada setiap submenu di dalam menu Dashboard, terdapat navbar pada bagian kiri untuk beralih ke submenu lainnya. Pada submenu Device, user dapat melihat nama device beserta informasi yang dimilikinya, yaitu Device ID, Latitude, dan Longitude. Di bawah informasi setiap device, terdapat *link* akses cepat menuju submenu Dashboard dan submenu Data dari device yang dipilih.

5. Submenu Dashboard



Gambar III.9 Submenu Dashboard

Pada submenu Dashboard, ditampilkan data terbaru dari masing-masing sensor IoT dan grafik dari masing-masing sensor IoT berdasarkan waktu. User dapat memilih device yang ingin ditampilkan pada dashboard dengan menekan tombol “Select Device” dan juga dapat mengatur rentang waktu dengan menekan tombol “Select Time Range on the Graph”. Hal ini memungkinkan user untuk memantau kondisi lingkungan secara *real-time* dan melihat tren historis dari setiap sensor IoT pada setiap device.

6. Submenu Data

The screenshot shows the LoraSense interface with a dark sidebar on the left containing 'Device', 'Dashboard', and 'Data' buttons. The main area is titled 'Data' and shows a table titled 'Data Table SF8'. The table has columns for Device ID, Time, Temperature, pH, TDS, DO, ORP, Salinity, Water Height, and Water Clarity. Each row contains data for a specific timestamp. A green 'Download Data' button is located at the top right of the table.

Device ID	Time	Temperature	pH	TDS	DO	ORP	Salinity	Water Height	Water Clarity
035303d90dc8ee94	2024-07-23 15:45:04	35.0°C	7.1	175.0 ppm	4.1 mg/L	450.5 mV	10.0 ppt	3.4 cm	6.2 NTU
035303d90dc8ee94	2024-07-23 15:44:59	35.0°C	7.1	175.0 ppm	4.1 mg/L	450.5 mV	10.0 ppt	3.4 cm	6.2 NTU
035303d90dc8ee94	2024-07-23 15:44:44	35.0°C	7.1	175.0 ppm	4.1 mg/L	450.5 mV	10.0 ppt	3.4 cm	6.2 NTU
035303d90dc8ee94	2024-07-23 15:44:39	34.0°C	7.1	175.0 ppm	4.1 mg/L	450.5 mV	10.0 ppt	3.4 cm	6.2 NTU
035303d90dc8ee94	2024-07-23 15:44:19	35.0°C	7.1	175.0 ppm	4.1 mg/L	450.5 mV	10.0 ppt	3.4 cm	6.2 NTU
035303d90dc8ee94	2024-07-23 15:44:14	35.0°C	7.1	175.0 ppm	4.1 mg/L	450.5 mV	10.0 ppt	3.4 cm	6.2 NTU
035303d90dc8ee94	2024-07-23 15:44:04	34.0°C	7.1	175.0 ppm	4.1 mg/L	450.5 mV	10.0 ppt	3.4 cm	6.2 NTU
035303d90dc8ee94	2024-07-23 15:43:54	35.0°C	7.1	175.0 ppm	4.1 mg/L	450.5 mV	10.0 ppt	3.4 cm	6.2 NTU
035303d90dc8ee94	2024-07-23 15:43:34	35.0°C	7.1	175.0 ppm	4.1 mg/L	450.5 mV	10.0 ppt	3.4 cm	6.2 NTU
035303d90dc8ee94	2024-07-23 15:43:29	35.0°C	7.1	175.0 ppm	4.1 mg/L	450.5 mV	10.0 ppt	3.4 cm	6.2 NTU

Gambar III.10 Submenu Data

Pada submenu Data, ditampilkan tabel data dari device yang dipilih dengan menekan tombol “Select Device”. Selain itu, terdapat tombol “Download Data” yang memungkinkan user untuk mengunduh data dari seluruh sensor IoT pada device yang telah dipilih. Fitur ini mempermudah user dalam mengakses dan menganalisis data secara mendalam, serta mendukung pengelolaan data dalam bentuk file yang dapat diakses secara offline.

7. Submenu Profile

The screenshot shows the LoraSense interface with a dark sidebar on the left containing 'Device', 'Dashboard', and 'Data' buttons. The main area is titled 'Profile' and shows a 'User Profile' section. It displays the Username (shafly) and Email (shafly@gmail.com). A green 'Edit User' button is located at the bottom of the profile section.

Gambar III.11 Submenu Profile

Untuk mengakses submenu Profile, user dapat menekan nama dan gambar user pada pojok kanan atas dashboard, lalu memilih opsi “Profile”. Pada submenu Profile, ditampilkan informasi user saat ini. Terdapat tombol “Edit User” yang memungkinkan user untuk mengedit informasi user seperti username, email, dan password. Fitur ini memberikan kemudahan bagi user untuk memperbarui dan mengelola informasi pribadi mereka agar tetap akurat dan terkini.

8. Submenu Admin Devices

Username	Device Name	Device ID	Latitude	Longitude	Edit Device	Delete Device
admin	SF10	909c98e2cf51c203	-6.99598	107.849	<button>Edit Device</button>	<button>Delete Device</button>
shafly	SF7	f8fffb291d63f3e57	-6.96088	107.791	<button>Edit Device</button>	<button>Delete Device</button>
shafly	SF8	035303d90dcBee94	-6.97341	107.81	<button>Edit Device</button>	<button>Delete Device</button>

Gambar III.12 Submenu Admin Devices

Submenu Admin Devices dan Admin Users hanya dapat diakses jika user yang login merupakan admin dari dashboard ini. Pada submenu Admin Devices, admin dapat melihat informasi semua device yang dimiliki oleh semua user. Admin dapat menambahkan device baru dengan menekan tombol “Add Device”, mengedit informasi device dengan menekan tombol “Edit Device”, dan menghapus device dengan menekan tombol “Delete Device”. Fitur-fitur tersebut memungkinkan admin untuk mengelola device secara efisien dan memastikan bahwa informasi device selalu terkini.

9. Submenu Admin Users

Username	User ID	Email	Is Admin
admin	1	admin@gmail.com	True
helmi	5	helmi123@gmail.com	False
shafly	3	shafly@gmail.com	False

Gambar III.13 Submenu Admin Users

Pada submenu Admin Users, admin dapat melihat seluruh informasi (kecuali kata sandi) dari setiap user yang telah melakukan registrasi. Informasi yang ditampilkan mencakup username, email, dan status admin. Submenu ini memungkinkan admin untuk mengelola user secara efektif.

3.2.2 Implementasi Backend

Dalam implementasi *backend*, penulis menggunakan *framework* Flask untuk menangani logika aplikasi web dan komunikasi dengan *frontend* dan database. Flask menyediakan berbagai *endpoint* API yang diperlukan untuk mengelola permintaan data dari *frontend*, pengambilan data dari database, dan pengiriman data

ke database. Untuk penyimpanan data, penulis menggunakan MySQL sebagai sistem manajemen basis data. Komunikasi antara Flask dan MySQL dilakukan melalui SQLAlchemy, yang berfungsi sebagai *Object Relational Mapper* (ORM).

3.2.2.1 Pengelolaan API dan Routing

1. Inisialisasi Aplikasi Flask

```
from flask import (
    Flask, g, request, render_template, redirect, url_for,
    make_response, send_file, flash
)
app = Flask(__name__)

# Load configuration from environment or default
app.config['SECRET_KEY'] = os.getenv('SECRET_KEY', 'default_secret_key')
```

Gambar III.14 Kode Program Inisiasi Aplikasi Flask

Aplikasi Flask diinisialisasi dan dikonfigurasi dengan pengaturan dasar, seperti SECRET_KEY, untuk tujuan keamanan. Penulis menempatkan SECRET_KEY dalam variabel di file .env untuk meningkatkan keamanan.

2. Pendefinisian Endpoint dan Routing

```
# Routes
@app.route('/')
def home():
    """Render the home page"""
    return render_template('home.html')

@app.route('/register', methods=['GET', 'POST'])
def register(): ...

@app.route('/login', methods=['GET', 'POST'])
def login(): ...

@app.route('/logout')
def logout(): ...

@app.route('/device')
@token_required
def device(): ...

@app.route('/dashboard', methods=['GET', 'POST'])
@token_required
def dashboard(): ...
```

Gambar III.15 Kode Program Pendefinisian Endpoint dan Routing

Setiap *endpoint* atau *route* dalam aplikasi ditentukan oleh decorator Flask dengan cara @app.route(). Setiap *endpoint* memiliki URL yang terkait dan metode HTTP (GET dan POST). *Endpoint* ini mengarahkan *request* user ke fungsi untuk mengakses menu atau fitur dari aplikasi. Berikut *endpoint* yang telah dibuat.

Tabel III.2 Endpoint API Aplikasi Flask dan Kegunaannya

Menu dan Fitur	Endpoint	HTTP Method	Kegunaan
Home	/	GET	Menampilkan halaman utama.
Login	/login	GET, POST	Menampilkan halaman login dan memproses login user.
Register	/register	GET, POST	Menampilkan halaman register dan memproses registrasi user baru.
Device	/device	GET	Menampilkan halaman device milik user.
Dashboard	/dashboard	GET, POST	Menampilkan dashboard device yang menunjukkan data terbaru dan grafik data terhadap waktu.
Data	/data	GET, POST	Menampilkan tabel yang berisi data dari device tertentu.
Profile	/profile	GET	Menampilkan halaman profil user.
Admin Devices	/admin-devices	GET	Menampilkan halaman manajemen device untuk admin.
Admin Users	/admin-users	GET	Menampilkan halaman manajemen user untuk admin.
Logout	/logout	GET	Melakukan logout user dan menghapus token dari cookie.
Download Data	/download_csv	POST	Mengunduh data dari device tertentu dalam format CSV.
Edit User	/update_user	POST	Memperbarui informasi user.
Add Device	/add_device	POST	Menambahkan device baru ke database oleh admin.
Edit Device	/update_device	POST	Memperbarui informasi device di database oleh admin.
Delete Device	/delete_device	POST	Menghapus device dari database oleh admin.

3. Middleware dan Error Handling

```

def generate_token(user_id):
    """ Generate a JWT token with the user ID"""
    expiration = datetime.utcnow() + timedelta(minutes=30)
    return jwt.encode(
        {'user_id': user_id, 'exp': expiration}, app.config['SECRET_KEY'], algorithm='HS256'
    )

def verify_token(token): ...

def token_required(f):
    """Decorator that ensures a valid token is present for the endpoint"""
    @wraps(f)
    def decorated(*args, **kwargs):
        token = request.cookies.get('token')
        if not token or not (decoded_token := verify_token(token)):
            return redirect(url_for('login'))
        g.current_user = decoded_token['user_id']
        return f(*args, **kwargs)
    return decorated

def admin_required(f):

```

Gambar III.16 Kode Program Middleware

```

# Error handler
@app.errorhandler(Exception)
def handle_exception(error):
    """Global error handler for handling exceptions"""
    app.logger.error(f"An error occurred: {error}")
    if request.referrer:
        return redirect(request.referrer)
    return redirect(url_for('home'))

```

Gambar III.17 Kode Program Error Handling

Middleware dalam aplikasi ini terdiri dari dua dekorator utama, yaitu `token_required` dan `admin_required`. Dekorator `token_required` memastikan bahwa *endpoint* yang dipilih hanya dapat diakses jika token JWT yang valid dimiliki oleh user. Jika token tidak ada atau tidak valid, user akan dialihkan ke halaman login. Dekorator `admin_required` memverifikasi bahwa user yang mengakses *endpoint* adalah seorang admin. Jika user bukan admin, user akan dialihkan kembali ke halaman sebelumnya. Untuk penanganan kesalahan (*error handling*), digunakan penanganan kesalahan global dengan fungsi `handle_exception` yang mencatat kesalahan yang terjadi dan mengarahkan user ke halaman sebelumnya atau ke halaman home jika halaman sebelumnya tidak ada. Penanganan kesalahan tersebut memastikan aplikasi tetap berjalan dengan baik meskipun terjadi kesalahan.

3.2.2.2 Manajemen Data dan Operasi Database

1. Koneksi dan Inisialisasi Database

```

from sqlalchemy import (
    create_engine, desc, column, Float, DateTime,
    Boolean, String, ForeignKey, Integer, or_
)
from sqlalchemy.orm import declarative_base, sessionmaker, scoped_session, relationship
from sqlalchemy.exc import IntegrityError
from dotenv import load_dotenv

# Load environment variables from the .env file
load_dotenv()

# Retrieve the variables from the environment
username = os.getenv('DB_USERNAME')
password = os.getenv('DB_PASSWORD')
hostname = os.getenv('DB_HOST')
port = os.getenv('DB_PORT')
database_name = os.getenv('DB_NAME')
app_lora = os.getenv('APP_LORA')
api_key = os.getenv('API_KEY_ANTARES')

# Construct the connection URL
connection_url = f'mysql+pymysql://{username}:{password}@{hostname}:{port}/{database_name}'

# Create the SQLAlchemy engine
engine = create_engine(connection_url)

```

Gambar III.18 Kode Program Koneksi dan Inisialisasi Database

Koneksi ke database dikelola menggunakan SQLAlchemy, sebuah ORM (*Object Relation Mapping*) yang memfasilitasi interaksi antara aplikasi Python dengan database SQL. Detail informasi koneksi, seperti username, password, hostname, port, dan database_name, disimpan dalam variabel environment di dalam file .env yang dapat dimuat menggunakan fungsi load_dotenv() untuk menjaga keamanan data sensitif. Pembuatan URL untuk koneksi database dilakukan dengan menggunakan format string yang menyertakan skema koneksi (mysql+pymysql), username, password, hostname, port, dan database_name. Setelah URL koneksi dibangun, SQLAlchemy *engine* dibuat menggunakan create_engine(connection_url), yang kemudian memungkinkan aplikasi untuk membuat *session* dan melakukan operasi database dengan mudah.

2. Pendefinisian Model Data

```

# Define the base class for declarative class definitions
Base = declarative_base()

class User(Base):
    """ ...
    __tablename__ = 'user'
    user_id = Column(Integer, primary_key=True, autoincrement=True)
    username = Column(String(100))
    email = Column(String(100))
    password = Column(String(100))
    admin = Column(Boolean, default=False)

    devices = relationship('Device', back_populates='user', cascade='all, delete-orphan')

    def set_password(self, password): ...
    def check_password(self, password): ...

class Device(Base):
    """ ...
    __tablename__ = 'device'
    device_id = Column(String(100), primary_key=True)
    device_name = Column(String(100))

```

Gambar III.19 Kode Program Pendefinisian Model Data

Model data dalam aplikasi didefinisikan menggunakan SQLAlchemy, yang menerjemahkan struktur objek Python menjadi tabel dalam database. Ada tiga kelas model yang didefinisikan yaitu tabel User, Device, dan Data. Tabel-tabel ini dibuat dengan `Base.metadata.create_all(engine)`, dan `session` database dikelola dengan sessionmaker serta scoped_session untuk interaksi yang efisien dan aman dengan database.

3. Fungsi CRUD dan Query

```
def add_device_to_db(user_id, device_id, device_name, latitude, longitude): ...
def update_device_to_db(old_device_id, new_user_id, new_device_id, new_device_name, new_latitude, new_longitude): ...
def delete_device_by_id(device_id):
    """Delete a device by its ID from the database."""
    session = Session()
    try:
        device = session.query(Device).filter_by(device_id=device_id).first()
        if device:
            session.delete(device)
            session.commit()
            return True
        return False
    except Exception as e:
        print(f"Error deleting device: {e}")
        session.rollback()
        return False
    finally:
        session.close()
```

Gambar III.20 Kode Program Fungsi CRUD

```
def get_user_id(username):
    """Retrieve user ID by username from the database."""
    session = Session()
    try:
        user = session.query(User).filter_by(username=username).first()
        return user.user_id
    finally:
        session.close()

def get_user_info(user_id):
    """Retrieve user information by user ID from the database."""
    session = Session()
    try:
        return session.query(User).filter_by(user_id=user_id).first()
    finally:
        session.close()
```

Gambar III.21 Kode Program Fungsi Query

Fungsi CRUD (*Create, Read, Update, Delete*) dibuat dalam *backend* untuk mengelola dan memodifikasi data di database. Fungsi CRUD dapat diaplikasikan untuk menambahkan data ke database seperti `add_device_to_db()`, mengambil data dari database seperti `get_device_data()`, mengedit data ke database seperti `update_device_to_db()`, dan menghapus data pada database seperti `delete_device_by_id()`. Sedangkan fungsi query memungkinkan pengambilan data berdasarkan parameter dan kondisi tertentu, untuk memberikan fleksibilitas dalam pengambilan data. Berikut fungsi CRUD dan query yang telah dibuat.

Tabel III.3 Daftar Fungsi CRUD dan Kegunaannya

Nama Fungsi	Kegunaan
add_user()	Menambahkan user baru ke database setelah memeriksa apakah username/email yang diinput belum terdaftar.
login_user()	Mengautentikasi login user berdasarkan username dan password.
user_exists()	Memeriksa apakah username atau email sudah ada di database.
is_admin()	Memeriksa apakah user memiliki hak akses admin.
update_user_info()	Memperbarui informasi user di database.
add_device_to_db()	Menambahkan device baru ke database setelah memeriksa apakah device dengan ID atau nama sudah ada.
update_device_in_db()	Memperbarui informasi device di database.
delete_device_by_id()	Menghapus device berdasarkan ID dari database.
device_exists()	Memeriksa apakah device dengan ID atau nama tertentu sudah ada di database.
get_all_devices()	Mengambil semua device dari database.
get_all_users()	Mengambil semua user dari database.
get_user_id()	Mengambil ID user berdasarkan username dari database.
get_user_info()	Mengambil informasi user berdasarkan ID dari database.
get_user_devices()	Mengambil device yang terkait dengan user tertentu dari database.
get_device_data()	Mengambil data entri untuk device tertentu dari database.
get_device_last_data()	Mengambil entri data terbaru untuk device tertentu dari database.
get_device_info()	Mengambil informasi device berdasarkan ID dari database.
get_data_by_device_and_time()	Mengambil entri data untuk device tertentu dalam rentang waktu tertentu dari database.
store_data()	Mengambil dan menyimpan data dari API eksternal ke database.

4. Integrasi dengan API Antares

```

55     def fetch_data_api_antares(device_id, url, headers, last_time):
56         """
57         # Send GET request with timeout
58         response = requests.get(url, headers=headers, timeout=10)
59         # Create an array to store the JSON objects
60         json_array = []
61         # Check if request was successful
62         if response.status_code == 200:
63             # Parse JSON response
64             data = response.json()
65             # Filter entries with the current date (only date part)
66             for entry in data["m2m:list"]:
67                 # Get the creation time
68                 creation_time_str = entry["m2m:cin"]["ct"]
69                 creation_time = datetime.strptime(creation_time_str, "%Y-%m-%dT%H:%M:%S")
70                 # Continue to the next entry if the creation time is before the last processed time
71                 if creation_time <= last_time:
72                     continue
73                 con_data = entry["m2m:cin"]["con"]
74                 # Check if the data is JSON
75                 try:
76                     con_data_json = json.loads(con_data)
77                     # Check if the data contains a "data" key
78                     if "data" in con_data_json:
79                         data_value = con_data_json["data"]
80                         # Check if data needs to be decompressed
81                         decompressed_data = decompress(data_value)
82                         # Create a new JSON object
83                         json_object = {
84                             "device_id": device_id,
85                             "url": url,
86                             "headers": headers,
87                             "last_time": last_time,
88                             "data": decompressed_data
89                         }
90                         # Add the JSON object to the array
91                         json_array.append(json_object)
92
93
94

```

Gambar III.22 Kode Program Fetch Data API Antares

Kode program `fetch_data_api_antares()` digunakan untuk mengambil data dari API Antares berdasarkan device_id, URL, dan header yang diberikan. Fungsi ini mengirim permintaan GET dan memeriksa status respons. Jika berhasil, respons JSON yang diterima diparsing dan difilter berdasarkan waktu terakhir data di database. Data yang valid lalu didekompresi dan dimasukkan ke dalam array objek JSON yang menjadi keluaran fungsi ini.

```
def decompress(data):
    """
    ...
    # Load the shared library (decompress library)
    lib = ctypes.CDLL('./libunishox2.dll') # Change to 'libunishox2.dll' on Windows

    # Define the Unishox2 decompress function signature
    lib.unishox2_decompress_simple.argtypes = [ctypes.c_char_p, ctypes.c_int, ctypes.c_char_p]
    lib.unishox2_decompress_simple.restype = ctypes.c_int

    compressed_data = bytes.fromhex(data)

    # Allocate a buffer for the decompressed data
    decompressed_data = ctypes.create_string_buffer(1024) # Adjust size as needed

    # Decompress the data
    decompressed_length = lib.unishox2_decompress_simple(
        compressed_data,
        len(compressed_data),
        decompressed_data
    )

    # Get the decompressed data as a string
    decompressed_string = decompressed_data.raw[:decompressed_length].decode('utf-8')

    # Convert the JSON string to a Python dictionary
    data_dict = json.loads(decompressed_string)

    return data_dict
```

Gambar III.23 Kode Program Dekompresi

Kode program `decompress()` digunakan untuk mendekompresi data dengan algoritma Unishox2. Cara kerja fungsi ini yaitu memuat *library* dekompresi, mengubah data heksadesimal menjadi bytes, dan mendekompresi data tersebut. Hasil dekompresi diubah menjadi string UTF-8 dan dikonversi menjadi bentuk *dictionary* Python sebagai keluaran fungsi.

```
def store_data(device_id):
    """
    Fetch and store data from an external API to the database.
    """
    session = Session()
    try:
        # Retrieve device name and last recorded time from the database
        device_name = get_device_info(device_id=device_id).device_name
        try:
            last_time = get_device_last_data(device_id=device_id).time
        except AttributeError:
            last_time = datetime(1970, 1, 1)

        # Define the URL and headers for the external API
        url = f"https://platform.antares.id:8443/~/antares-cse/antares-id/{app_lora}/{device_name}?fu=1&drt=2&ty=4"
        headers = {
            "X-M2M-Origin": f"{api_key}",
            "Content-Type": "application/json",
            "Accept": "application/json"
        }

        # Fetch data from the API
        json_array = fetch_data_api_antares(device_id, url, headers, last_time)
```

Gambar III.24 Kode Program Store Data

Kode program `store_data()` digunakan untuk mengambil data dari API Antares dan menyimpannya ke dalam database. Fungsi ini mengambil informasi device dan waktu data terbaru dari database, mengambil data baru dari API menggunakan fungsi `fetch_data_api_antares`, dan menyimpan data tersebut ke database, dengan memastikan data yang baru ditambahkan tidak menyebabkan konflik dengan data yang sudah ada pada database.

BAB IV PENGUJIAN DAN ANALISIS

4.1 Pengujian Desain

4.1.1 Pengujian Desain Frontend

Pengujian desain *frontend* didasari pada subobjektif 8, yaitu *user interface* yang mudah digunakan dengan syarat pemenuhan sesuai standar ISO 9241. ISO 9241 merupakan pedoman yang mengatur syarat-syarat umum dalam merancang dan menilai tampilan visual elektronik. Pedoman ini meliputi berbagai jenis tampilan, seperti monitor komputer, layar, dan perangkat visual elektronik lainnya. Pengujian *user interface* (UI) yang mudah digunakan sangat penting karena dapat memastikan bahwa desain sederhana dan dapat diakses oleh berbagai usia, serta dapat memenuhi kebutuhan user.

Dalam pengujian ini, user diminta untuk menjalankan dashboard menggunakan browser melalui alamat IP Public <http://18.141.202.40/>. Kemudian user dapat menilai dashboard tersebut dengan mengisi kuesioner berdasarkan pertanyaan-pertanyaan berikut, menggunakan skala likert:

1. Seberapa mudah navigasi pada dashboard ini?
2. Apakah informasi dari data yang ditampilkan mudah dipahami?
3. Seberapa cepat dashboard memuat data?
4. Seberapa cocok dashboard tersebut digunakan untuk pemantauan data sensor IoT?

Pertanyaan 1 digunakan untuk mengukur kemudahan penggunaan (*usability*) dashboard, ditunjukkan dengan kemampuan mengakses berbagai menu, submenu, dan fitur yang ada. Pertanyaan 2 digunakan untuk menilai kejelasan (*clarity*) dan efektifitas desain, yaitu sejauh mana informasi yang ditampilkan melalui tabel, grafik, atau elemen visual lainnya mudah dipahami oleh pengguna. Pertanyaan 3 digunakan untuk menilai kecepatan *loading* berdasarkan pengalaman pengguna, yaitu seberapa cepat data ditampilkan setelah *request* dilakukan pada dashboard. Pertanyaan 4 digunakan untuk menilai relevansi dan fungsionalitas dashboard, apakah sesuai dengan tujuan dan spesifikasi yang dibutuhkan untuk pemantauan data sensor IoT.

Secara umum, skala likert merupakan skala yang digunakan untuk menilai suatu objek berdasarkan pendapat individu atau kelompok dengan menggunakan teknis dan definisi operasional yang telah ditetapkan peneliti.

Skor skala likert dapat dihitung menggunakan persamaan sebagai berikut:

$$skor \text{ skala likert} = T \times Pn \quad (1)$$

Dengan T merupakan jumlah responden yang memilih skor tertentu dan Pn merupakan pilihan angka skor likert. Skor skala likert yang telah didapatkan untuk masing-masing pilihan angka skor likert dapat dijumlahkan sehingga dihasilkan nilai total skor.

Dari nilai total skor yang telah dihasilkan, dapat ditentukan indeks skala likert menggunakan persamaan sebagai berikut:

$$indeks \text{ skala likert} (\%) = \frac{total \text{ skor}}{Y} \times 100 \quad (2)$$

Dengan nilai Y sebagai berikut:

$$Y = skor \text{ likert tertinggi yang digunakan} \times jumlah \text{ responden} \quad (3)$$

Indeks skala likert yang telah ditentukan dapat diklasifikasikan sebagai berikut:

Tabel IV.1 Klasifikasi Indeks Skala Likert

Indeks skala likert (%)	Klasifikasi
0-19,99	Sangat tidak setuju
20-39,99	Tidak setuju
40-59,99	Netral
60-79,99	Setuju
80-100	Sangat setuju

Rentang interval untuk mengategorikan indeks skala likert didapatkan dengan persamaan sebagai berikut:

$$Interval (\%) = \frac{100}{\text{banyaknya pilihan angka skor likert}} \quad (4)$$

Penilaian *user interface* yang dilakukan memiliki skor likert dari 1 hingga 5. Skor 1 menunjukkan user sangat tidak setuju, sedangkan skor 5 menunjukkan user sangat setuju. Berikut ini adalah hasil dari kuesioner tersebut.

Tabel IV.2 Hasil Kuesioner Penilaian User Interface

Pertanyaan	Skor Likert	Jumlah Responden	Percentase (%)
Seberapa mudah navigasi pada dashboard ini?	1	0	0
	2	0	0
	3	1	2.2
	4	9	20
	5	35	77.8
Apakah informasi dari data yang ditampilkan mudah dipahami?	1	0	0
	2	1	2.2
	3	5	11.1
	4	15	33.3
	5	24	53.3
Seberapa cepat dashboard memuat data?	1	0	0
	2	0	0
	3	3	6.7
	4	14	31.1
	5	28	62.2
Seberapa cocok dashboard tersebut digunakan untuk pemantauan data sensor IoT?	1	0	0
	2	0	0
	3	3	6.7
	4	20	44.4
	5	22	48.9

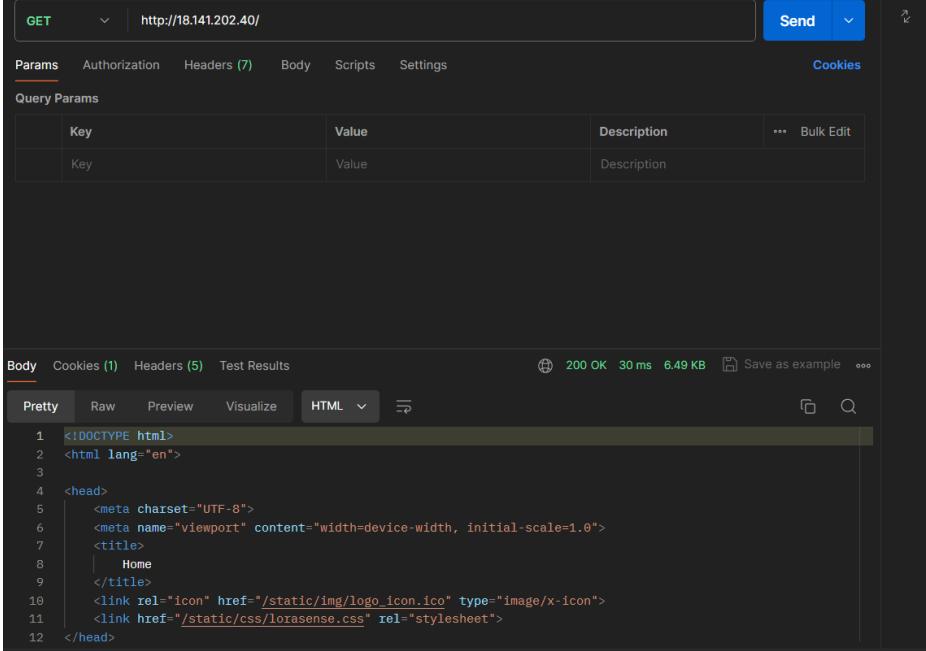
4.1.2 Pengujian Desain Backend

Pengujian desain *backend* didasari pada subobjektif 6, yaitu integrasi data yang akurat dan *real-time*, serta subobjektif 7, yaitu penyediaan data *backup*. Subobjektif 6 dapat dicapai dengan memastikan data dari sensor diterima dan ditampilkan pada dashboard dengan *latency* rendah. Sedangkan subobjektif 7 dapat dicapai dengan mengimplementasikan mekanisme *backup* otomatis setiap interval waktu tertentu untuk memastikan data sensor IoT yang telah didekompresi selalu tersedia dan tidak hilang.

4.1.2.1 Pengujian Kinerja Web Dashboard

Untuk memenuhi subobjektif 6, dilakukan pengujian terhadap kinerja *web dashboard* yang dapat melakukan *request* dan menerima respons dari *backend*. Pada *web dashboard* yang telah dibuat, dilakukan pengujian pada setiap menu, submenu, dan fitur di dalamnya.

1. Menu Home



The screenshot shows the Postman application interface. At the top, there is a header with 'GET' and the URL 'http://18.141.202.40/'. Below the header, there are tabs for 'Params', 'Authorization', 'Headers (7)', 'Body', 'Scripts', and 'Settings'. On the right side of the header, there is a 'Send' button and a dropdown menu. Under the 'Params' tab, there is a section titled 'Query Params' with a table. The table has columns for 'Key' and 'Value', with one row containing 'Key' and 'Value'. Below the table, there are tabs for 'Body', 'Cookies (1)', 'Headers (5)', and 'Test Results'. The 'Body' tab is selected, showing the raw HTML code of the response. The response status is displayed as '200 OK 30 ms 6.49 KB'. The HTML code includes doctype, meta tags for charset and viewport, a title with the text 'Home', and links to logo icons and CSS files. At the bottom of the interface, there are various buttons and icons for managing the request.

Gambar IV.1 Pengujian Menu Home

Pengujian menu home dengan mengakses URL <http://18.141.202.40/> menggunakan metode GET berhasil dilakukan. Terdapat latensi sebesar 30 ms dalam mengakses URL tersebut.

2. Menu Login

The screenshot shows the Postman interface with a POST request to <http://18.141.202.40/login>. The 'Body' tab is selected, showing a table with two rows: 'username' with value 'shafly' and 'password' with value 'shafly123'. The 'Headers' tab shows 9 headers. The 'Test Results' tab displays the HTML response code, which includes meta tags for viewport, description, and author, along with links for logo icon and CSS files.

Gambar IV.2 Pengujian Menu Login

Pengujian menu login dengan mengakses URL <http://18.141.202.40/login> menggunakan metode POST berhasil dilakukan dengan Body berisi *key-value* sebagai berikut:

- ‘username’ : ‘shafly’
- ‘password’ : ‘shafly123’

Terdapat latensi sebesar 40 ms dalam mengakses URL tersebut.

3. Menu Register

The screenshot shows the Postman interface with a POST request to <http://18.141.202.40/register>. The 'Body' tab is selected, showing a table with four rows: 'username' with value 'user1', 'email' with value 'user1@gmail.com', 'password' with value 'User123', and 'password2' with value 'User123'. The 'Headers' tab shows 7 headers. The 'Test Results' tab displays the HTML response code, which includes a DOCTYPE declaration, meta tags for charset and viewport, a title 'Login', and links for logo icon and CSS files.

Gambar IV.3 Pengujian Menu Register

Pengujian menu register dengan mengakses URL <http://18.141.202.40/register> menggunakan metode POST berhasil dilakukan dengan Body berisi *key-value* sebagai berikut:

- ‘username’ : ‘user1’
- ‘email’ : ‘user1@gmail.com’
- ‘password’ : ‘User123’
- ‘password2’ : ‘User123’.

Terdapat latensi sebesar 35 ms dalam mengakses URL tersebut.

4. Submenu Device

The screenshot shows the Postman application interface. At the top, there is a header bar with 'GET' selected, the URL 'http://18.141.202.40/device', and a 'Send' button. Below the header are tabs for 'Params', 'Authorization', 'Headers (7)', 'Body', 'Scripts', and 'Settings'. The 'Headers' tab is active, showing a table with one row: 'Content-Type' set to 'application/json'. The 'Body' tab is also visible. On the right side, there are tabs for 'Cookies' and 'Query Params', both of which are empty. At the bottom of the interface, there are tabs for 'Body', 'Cookies (1)', 'Headers (5)', and 'Test Results'. The 'Test Results' tab is active, displaying a status of '200 OK' with a latency of '39 ms' and a size of '11.48 KB'. Below this, the raw HTML response is shown, starting with meta tags and a title element containing the word 'Device'. The code is numbered from 8 to 19. At the very bottom of the interface, there are several small icons for 'Postbot', 'Runner', 'Start Proxy', 'Cookies', 'Vault', 'Trash', and a help icon.

Gambar IV.4 Pengujian Submenu Device

Pengujian submenu device dengan mengakses URL <http://18.141.202.40/device> menggunakan metode GET berhasil dilakukan. Terdapat latensi sebesar 39 ms dalam mengakses URL tersebut.

5. Submenu Dashboard

The screenshot shows a POSTMAN interface with a GET request to http://18.141.202.40/dashboard?device_id=035303d90dc8ee94&time_range=this_week. The Params tab displays two key-value pairs: device_id (035303d90dc8ee94) and time_range (this_week). The Body tab shows the raw HTML response, which includes meta tags, a title tag with 'Dashboard', and links to logo icons and CSS files.

```

8 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
9 <meta name="description" content="">
10 <meta name="author" content="">
11
12 <title>
13 | Dashboard
14 </title>
15 <link rel="icon" href="/static/img/logo_icon.ico" type="image/x-icon">
16
17 <!-- Custom fonts -->
18 <link href="/static/vendor/fontawesome-free/css/all.min.css" rel="stylesheet" type="text/css">
19 <link

```

Gambar IV.5 Pengujian Submenu Dashboard

Pengujian submenu dashboard dengan mengakses URL <http://18.141.202.40/dashboard> menggunakan metode GET berhasil dilakukan dengan Params berisi *key-value* sebagai berikut:

- ‘device_id : ‘035303d90dc8ee94’
- ‘time_range’ : ‘this_week’

Terdapat latensi sebesar 133 ms dalam mengakses URL tersebut.

6. Submenu Data

The screenshot shows a POSTMAN interface with a GET request to http://18.141.202.40/data?device_id=035303d90dc8ee94. The Params tab displays one key-value pair: device_id (035303d90dc8ee94). The Body tab shows the raw HTML response, which includes meta tags, a title tag with 'Data', and a link to a logo icon.

```

8 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
9 <meta name="description" content="">
10 <meta name="author" content="">
11
12 <title>
13 | Data
14 </title>
15 <link rel="icon" href="/static/img/logo_icon.ico" type="image/x-icon">

```

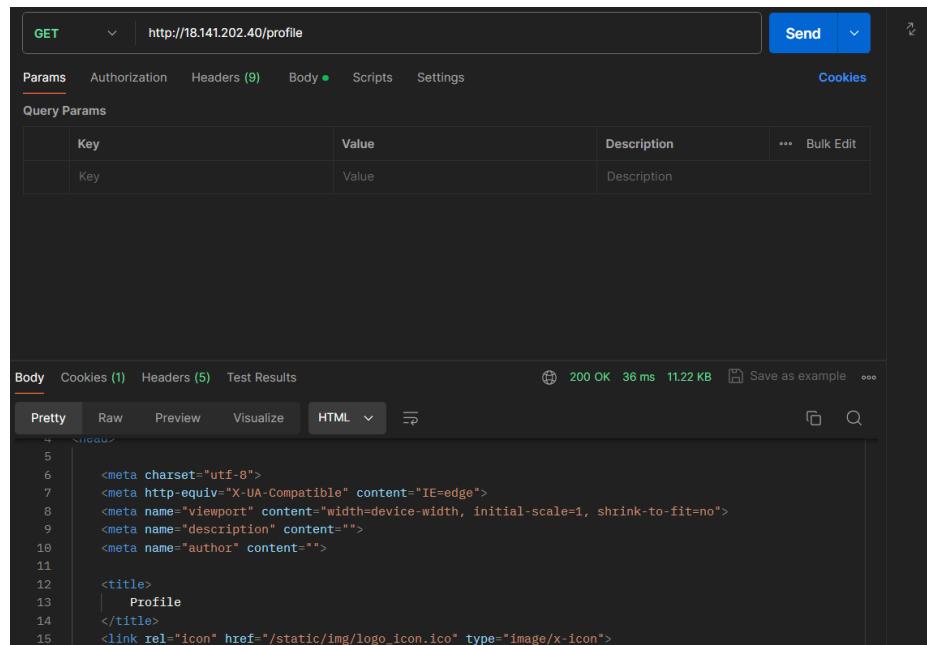
Gambar IV.6 Pengujian Submenu Data

Pengujian submenu data dengan mengakses URL <http://18.141.202.40/data> menggunakan metode GET berhasil dilakukan dengan Params berisi *key-value* sebagai berikut:

- ‘device_id : ‘035303d90dc8ee94’

Terdapat latensi sebesar 132 ms dalam mengakses URL tersebut.

7. Submenu Profile



The screenshot shows the Postman application interface. At the top, it displays a 'GET' method and the URL 'http://18.141.202.40/profile'. Below the URL, there are tabs for 'Params', 'Authorization', 'Headers (9)', 'Body', 'Scripts', and 'Settings'. The 'Params' tab is selected, showing a table with one row: 'Key' and 'Value'. In the 'Body' tab, the response is displayed as a snippet of HTML code, which includes meta tags for charset, viewport, and description, followed by a title tag containing 'Profile' and a link tag for an icon.

```
4 <!-->
5 <meta charset="utf-8">
6 <meta http-equiv="X-UA-Compatible" content="IE=edge">
7 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
8 <meta name="description" content="">
9 <meta name="author" content="">
10
11
12 <title>
13 | Profile
14 </title>
15 <link rel="icon" href="/static/img/logo_icon.ico" type="image/x-icon">
```

Gambar IV.7 Pengujian Submenu Profile

Pengujian submenu profile dengan mengakses URL <http://18.141.202.40/profile> menggunakan metode GET berhasil dilakukan. Terdapat latensi sebesar 132 ms dalam mengakses URL tersebut.

8. Submenu Admin Devices

GET | http://18.141.202.40/admin-devices

Params Authorization Headers (7) Body Scripts Settings Cookies

Key	Value	Description	... Bulk Edit
Key	Value	Description	

Body Cookies (1) Headers (5) Test Results

Pretty Raw Preview Visualize HTML

```

7 <meta http-equiv="X-UA-Compatible" content="IE=edge">
8 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
9 <meta name="description" content="">
10 <meta name="author" content="">
11
12 <title>
13 Admin Device
14 </title>
15 <link rel="icon" href="/static/img/logo_icon.ico" type="image/x-icon">
16
17 <!-- Custom fonts -->

```

200 OK 69 ms 31.92 KB Save as example ...

Postbot Runner Start Proxy Cookies Vault Trash

Gambar IV.8 Pengujian Submenu Admin Devices

Pengujian submenu admin devices dengan mengakses URL <http://18.141.202.40/admin-devices> menggunakan metode GET berhasil dilakukan. Terdapat latensi sebesar 69 ms dalam mengakses URL tersebut.

9. Submenu Admin Users

GET | http://18.141.202.40/admin-users

Params Authorization Headers (7) Body Scripts Settings Cookies

Key	Value	Description	... Bulk Edit
Key	Value	Description	

Body Cookies (1) Headers (5) Test Results

Pretty Raw Preview Visualize HTML

```

158
159
160 <!-- Main Content Section -->
161 <div class="container-fluid">
162
163 <!-- Page Heading -->
164 <div class="d-sm-flex align-items-center justify-content-between mb-4">
165 | <h1 class="h3 mb-0 text-gray-800">Admin Users</h1>
166 </div>
167
168 <!-- DataTales Example -->
169 <div class="card shadow mb-4">

```

200 OK 45 ms 11.26 KB Save as example ...

Postbot Runner Start Proxy Cookies Vault Trash

Gambar IV.9 Pengujian Submenu Admin Users

Pengujian submenu admin users dengan mengakses URL <http://18.141.202.40/admin-users> menggunakan metode GET berhasil dilakukan. Terdapat latensi sebesar 45 ms dalam mengakses URL tersebut.

10. Fitur Logout

The screenshot shows a Postman interface with the following details:

- Method: POST
- URL: http://18.141.202.40/logout
- Headers: (8)
- Body: (Pretty) HTML response showing the logout page source code.
- Test Results: Status: 200 OK, Time: 50 ms, Size: 6.49 KB

Gambar IV.10 Pengujian Fitur Logout

Pengujian fitur logout dengan mengakses URL <http://18.141.202.40/logout> menggunakan metode POST berhasil dilakukan. Terdapat latensi sebesar 50 ms dalam mengakses URL tersebut.

11. Fitur Download Data

The screenshot shows a Postman interface with the following details:

- Method: POST
- URL: http://18.141.202.40/download_csv
- Headers: (9)
- Body: (x-www-form-urlencoded)
 - selected_device_id: 035303d90dc8ee94
 - username: shafly
 - password: shafly123
- Test Results: Status: 200 OK, Time: 33 ms, Size: 3.29 KB

Gambar IV.11 Pengujian Fitur Download Data

Pengujian fitur download data dengan mengakses URL http://18.141.202.40/download_csv menggunakan metode POST berhasil dilakukan dengan Body berisi *key-value* sebagai berikut:

- ‘selected_device_id’ : ‘035303d90dc8ee94’

Terdapat latensi sebesar 33 ms dalam mengakses URL tersebut.

12. Fitur Edit User

The screenshot shows the Postman application interface. A POST request is being made to the URL http://18.141.202.40/update_user. The 'Body' tab is selected, showing the following key-value pairs:

Key	Value	Description	...	Bulk Edit
selected_device_id	035303d90dc8ee94			
username	shafly			
email	shafly456@gmail.com			
password	shafly456			
password2	shafly456			

Below the table, the 'Pretty' tab is selected, showing the raw HTML response from the server:

```
8 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
9 <meta name="description" content="">
10 <meta name="author" content="">
11
12 <title>
13 | Profile
14 </title>
15 <link rel="icon" href="/static/img/logo_icon.ico" type="image/x-icon">
16
17 <!-- Custom fonts -->
18 <link href="/static/vendor/fontawesome-free/css/all.min.css" rel="stylesheet" type="text/css">
19 <link
```

Gambar IV.12 Pengujian Fitur Edit User

Pengujian fitur edit user dengan mengakses URL http://18.141.202.40/update_user menggunakan metode POST berhasil dilakukan dengan Body berisi *key-value* sebagai berikut:

- ‘username’ : ‘shafly’
- ‘email’ : ‘shafly456@gmail.com’
- ‘password’ : ‘shafly456’
- ‘password2’ : ‘shafly456’

Terdapat latensi sebesar 36 ms dalam mengakses URL tersebut.

13. Fitur Add Device

POST http://18.141.202.40/add_device

Params Authorization Headers (9) **Body** Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

<input checked="" type="checkbox"/> password2	shafly123
<input checked="" type="checkbox"/> username	shafly
<input checked="" type="checkbox"/> device_id	1c54d1ba285b488d
<input checked="" type="checkbox"/> device_name	Device Shafly
<input checked="" type="checkbox"/> latitude	-6.99698
<input checked="" type="checkbox"/> longitude	107.87654
Key	Value
	Description

Body Cookies (1) Headers (7) Test Results

Pretty Raw Preview Visualize HTML

```

6   <meta charset="utf-8">
7   <meta http-equiv="X-UA-Compatible" content="IE=edge">
8   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
9   <meta name="description" content="">
10  <meta name="author" content="">

```

Gambar IV.13 Pengujian Fitur Add Device

Pengujian fitur add device dengan mengakses URL http://18.141.202.40/add_device menggunakan metode POST berhasil dilakukan dengan Body berisi *key-value* sebagai berikut:

- ‘username’ : ‘shafly’
- ‘device_id’ : ‘1c54d1ba285b488d’
- ‘device_name’ : ‘Device Shafly’
- ‘latitude’ : ‘-6.99698’
- ‘longitude’ : ‘107.87654’

Terdapat latensi sebesar 99 ms dalam mengakses URL tersebut.

14. Fitur Edit Device

POST http://18.141.202.40/update_device

Params Authorization Headers (9) **Body** Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

<input checked="" type="checkbox"/> password2	shafly123
<input checked="" type="checkbox"/> username	shafly
<input checked="" type="checkbox"/> device_id	909c98e2cf51c203
<input checked="" type="checkbox"/> device_name	SF10
<input checked="" type="checkbox"/> latitude	-6.99598
<input checked="" type="checkbox"/> longitude	107.845
Key	Value
	Description

Body Cookies (1) Headers (7) Test Results

Pretty Raw Preview Visualize HTML

```

5   <meta>
6   <meta charset="utf-8">
7   <meta http-equiv="X-UA-Compatible" content="IE=edge">
8   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
9   <meta name="description" content="">
10  <meta name="author" content="">
11  <title>
12  | Admin Device
13

```

Gambar IV.14 Pengujian Fitur Edit Device

Pengujian fitur edit device dengan mengakses URL http://18.141.202.40/update_device menggunakan metode POST berhasil dilakukan dengan Body berisi *key-value* sebagai berikut:

- ‘username’ : ‘shafly’
- ‘device_id’ : ‘909c98e2cf51c203’
- ‘device_name’ : ‘SF10’
- ‘latitude’ : ‘-6.9958’
- ‘longitude’ : ‘107.845’

Terdapat latensi sebesar 96 ms dalam mengakses URL tersebut.

15. Fitur Delete Device

The screenshot shows a Postman interface with a POST request to http://18.141.202.40/delete_device. The 'Body' tab is selected, showing the following key-value pairs in 'x-www-form-urlencoded' format:

Key	Value
password2	shafly123
username	shafly
<input checked="" type="checkbox"/> device_id	1c54d1ba285b488d
<input type="checkbox"/> device_name	Device Shafly
<input type="checkbox"/> latitude	-6.99698
<input type="checkbox"/> longitude	107.87654

The response tab shows the following HTML content:

```
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<meta name="description" content="">
<meta name="author" content="">
<title>
    Admin Device
</title>
<link rel="icon" href="/static/img/logo_icon.ico" type="image/x-icon">
```

Gambar IV.15 Pengujian Fitur Delete Device

Pengujian fitur delete device dengan mengakses URL http://18.141.202.40/delete_device menggunakan metode POST berhasil dilakukan dengan Body berisi *key-value* sebagai berikut:

- ‘device_id’ : ‘1c54d1ba285b488d’

Terdapat latensi sebesar 85 ms dalam mengakses URL tersebut.

4.1.2.2 Pengujian Backup Data

Untuk memenuhi subobjektif 7, pengujian *backup* data dilakukan untuk memastikan bahwa mekanisme *backup* data otomatis berjalan dengan baik dan data sensor IoT tetap aman serta tersedia.

Ketika user melakukan akses login, lalu beralih ke submenu dashboard, Flask secara otomatis akan menjalankan fungsi `store_data()`. Fungsi ini akan mengambil data terbaru dari Antares dan membandingkannya dengan data terbaru yang ada di database MySQL. Jika data terbaru dari Antares lebih terkini daripada data terbaru yang ada di database MySQL, maka data yang lebih baru dari Antares akan ditambahkan ke database MySQL. Dengan demikian, proses *backup* data akan dilakukan setiap kali user mengakses submenu dashboard. Berikut hasil pengujian fungsi `store_data()`.

```
PS D:\Penyimpanan Utama\Desktop\Tugas Akhir> python
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun 6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from database import store_data
>>> store_data('035303d90dc8ee94')
Successfully fetched 39 data points from device 035303d90dc8ee94.
Data successfully stored in the database.
>>> █
```

Gambar IV.16 Pengujian Store Data

Fungsi `store_data()` tersebut berhasil mengambil 39 entri data milik device dengan Device ID ‘035303d90dc8ee94’ dari Antares dan menyimpannya ke database MySQL. Berikut hasil dari penyimpanan data pada database MySQL.

	data_id	device_id	time	temp	ph	tds	do	orp	salinity	water_h	water_cl
1	133	8ff8291d63f3e57	2024-07-23 16:08:47	34	7.1	175	4.1	450.5	10	3.4	6.2
2	134	8ff8291d63f3e57	2024-07-23 16:07:37	34	7.1	175	4.1	450.5	10	3.4	6.2
3	135	8ff8291d63f3e57	2024-07-23 16:06:08	34	7.1	175	4.1	450.5	10	3.4	6.2
4	136	8ff8291d63f3e57	2024-07-23 16:06:03	34	7.1	175	4.1	450.5	10	3.4	6.2
5	137	8ff8291d63f3e57	2024-07-23 16:05:18	34	7.1	175	4.1	450.5	10	3.4	6.2
6	138	8ff8291d63f3e57	2024-07-23 16:04:37	34	7.1	175	4.1	450.5	10	3.4	6.2
7	139	8ff8291d63f3e57	2024-07-23 16:03:48	33	7.1	175	4.1	450.5	10	3.4	6.2
8	140	8ff8291d63f3e57	2024-07-23 16:03:28	33	7.1	175	4.1	450.5	10	3.4	6.2
9	141	8ff8291d63f3e57	2024-07-23 16:03:03	33	7.1	175	4.1	450.5	10	3.4	6.2
10	142	8ff8291d63f3e57	2024-07-23 16:02:18	33	7.1	175	4.1	450.5	10	3.4	6.2
11	143	8ff8291d63f3e57	2024-07-23 15:59:39	33	7.1	175	4.1	450.5	10	3.4	6.2
12	144	8ff8291d63f3e57	2024-07-23 15:58:54	34	7.1	175	4.1	450.5	10	3.4	6.2
13	145	8ff8291d63f3e57	2024-07-23 15:57:40	33	7.1	175	4.1	450.5	10	3.4	6.2
14	146	8ff8291d63f3e57	2024-07-23 15:56:35	33	7.1	175	4.1	450.5	10	3.4	6.2
15	147	8ff8291d63f3e57	2024-07-23 15:55:25	33	7.1	175	4.1	450.5	10	[NULL]	[NULL]
16	148	8ff8291d63f3e57	2024-07-23 15:53:55	33	7.1	175	4.1	450.5	10	[NULL]	[NULL]
17	149	8ff8291d63f3e57	2024-07-23 15:53:06	33	7.1	175	4.1	[NULL]	[NULL]	[NULL]	[NULL]
18	150	8ff8291d63f3e57	2024-07-23 15:51:36	33	7.1	175	4.1	[NULL]	[NULL]	[NULL]	[NULL]
19	151	8ff8291d63f3e57	2024-07-23 15:50:47	33	7.1	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]
20	152	8ff8291d63f3e57	2024-07-23 15:50:02	33	7.1	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]
21	153	8ff8291d63f3e57	2024-07-23 15:49:37	33	7.1	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]
22	154	8ff8291d63f3e57	2024-07-23 15:49:17	33	7.1	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]
23	155	035303d90dc8ee94	2024-07-23 15:45:04	35	7.1	175	4.1	450.5	10	3.4	6.2
24	156	035303d90dc8ee94	2024-07-23 15:44:59	35	7.1	175	4.1	450.5	10	3.4	6.2
25	157	035303d90dc8ee94	2024-07-23 15:44:44	35	7.1	175	4.1	450.5	10	3.4	6.2
26	158	035303d90dc8ee94	2024-07-23 15:44:39	34	7.1	175	4.1	450.5	10	3.4	6.2
27	159	035303d90dc8ee94	2024-07-23 15:44:19	35	7.1	175	4.1	450.5	10	3.4	6.2
28	160	035303d90dc8ee94	2024-07-23 15:44:14	35	7.1	175	4.1	450.5	10	3.4	6.2
29	161	035303d90dc8ee94	2024-07-23 15:44:04	34	7.1	175	4.1	450.5	10	3.4	6.2
30	162	035303d90dc8ee94	2024-07-23 15:43:54	35	7.1	175	4.1	450.5	10	3.4	6.2
31	163	035303d90dc8ee94	2024-07-23 15:43:34	35	7.1	175	4.1	450.5	10	3.4	6.2

Gambar IV.17 Pengujian Backup Data

Gambar di atas menunjukkan penyimpanan data yang berhasil dilakukan pada database MySQL. Data yang diambil dari Antares secara otomatis akan disimpan di dalam tabel Data dengan menambahkan baris baru. Tabel tersebut menyimpan semua data dari seluruh device yang terdapat pada tabel Device. Penambahan device pada tabel Device dilakukan secara manual oleh admin.

4.2 Analisis Hasil Pengujian

4.2.1 Analisis Hasil Pengujian Desain Frontend

Berdasarkan pengujian desain *frontend*, didapatkan hasil kuesioner pada Tabel IV.2. Dari tabel tersebut, dapat diperoleh skor skala likert dari masing-masing skor pada setiap pertanyaan menggunakan persamaan (1). Skor skala likert tersebut dijumlahkan sehingga menghasilkan nilai total skor. Berdasarkan persamaan (4), dihasilkan nilai parameter Y sebesar 225. Dengan diketahuinya nilai parameter Y dan total skor, dapat dihitung nilai indeks skala likert menggunakan persamaan (3) sebagai berikut.

Tabel IV.3 Analisis Hasil Kuesioner Penilaian User Interface

Pertanyaan	Skor Likert	Jumlah Responden	Skor Skala Likert	Total Skor	Indeks Skala Likert(%)
Seberapa mudah navigasi pada dashboard ini?	1	0	0	214	95.11
	2	0	0		
	3	1	3		
	4	9	36		
	5	35	175		
Apakah informasi dari data yang ditampilkan mudah dipahami?	1	0	0	197	87.55
	2	1	2		
	3	5	15		
	4	15	60		
	5	24	120		
Seberapa cepat dashboard memuat data?	1	0	0	205	91.11
	2	0	0		
	3	3	9		
	4	14	56		
	5	28	140		
Seberapa cocok dashboard tersebut digunakan untuk pemantauan data sensor IoT?	1	0	0	199	88.44
	2	0	0		
	3	3	9		
	4	20	80		
	5	22	110		

Hasil dari pertanyaan 1, didapatkan nilai indeks skala likert sebesar 95.11%, menunjukkan bahwa navigasi pada dashboard yang telah dibuat “Sangat Mudah”.

Hasil dari pertanyaan 2, didapatkan nilai indeks skala likert sebesar 87.55%, menunjukkan bahwa infomasi dari data yang ditampilkan “Sangat Mudah” untuk dipahami.

Hasil dari pertanyaan 3, didapatkan nilai indeks skala likert sebesar 91.11%, menunjukkan dashboard memuat data dengan “Sangat Cepat”.

Hasil dari pertanyaan 4, didapatkan nilai indeks skala likert sebesar 88.44%, menunjukkan dashboard tersebut “Sangat Cocok” digunakan untuk pemantauan data sensor IoT.

Berdasarkan analisis hasil kuesioner pada pertanyaan 1 dan 2, dashboard yang telah dibuat sangat mudah untuk dilakukan navigasi dan data yang ditampilkan sangat mudah dipahami. Oleh karena itu, dapat disimpulkan bahwa subobjektif 8 yaitu *user interface* yang mudah digunakan sudah berhasil terpenuhi.

4.2.2 Analisis Hasil Pengujian Desain Backend

Berdasarkan pengujian desain *backend*, didapatkan hasil kinerja *web dashboard* sebagai berikut.

Tabel IV.4 Analisis Hasil Pengujian Kinerja Web Dashboard

Menu dan Fitur	Status	Latensi (ms)
Home	Success	30
Login	Success	40
Register	Success	35
Device	Success	39
Dashboard	Success	133
Data	Success	132
Profile	Success	36
Admin Devices	Success	69
Admin Users	Success	45
Logout	Success	50
Download Data	Success	33
Edit User	Success	36
Add Device	Success	99
Edit Device	Success	96
Delete Device	Success	85

Berdasarkan Tabel IV.4, semua menu, submenu, dan fitur sudah berhasil dijalankan dengan kode status HTTP “200 OK”, yang menunjukkan bahwa *request* berhasil dipenuhi dan server mengirimkan respons yang diharapkan. Latensi untuk menu, submenu, dan fitur bervariasi dari 30 ms hingga 133 ms.

Latensi tertinggi yaitu sebesar 133 ms untuk submenu dashboard. Latensi yang tinggi ini disebabkan karena pada submenu tersebut terdapat proses pengambilan data dari database MySQL, pengecekan data pada database MySQL dengan data di Antares, serta menampilkan banyak grafik dari data yang tersimpan.

Untuk aplikasi *real-time* seperti *video conferencing* atau game online, latensi di bawah 50 ms dianggap ideal. Sedangkan untuk aplikasi web, latensi di bawah 100 ms dianggap baik, sementara latensi di antara 100 ms hingga 200 ms masih dapat diterima. Latensi di atas 200 ms dianggap buruk dan dapat memengaruhi performa aplikasi web secara signifikan.

Dalam konteks penelitian ini, aplikasi web digunakan hanya untuk pemantauan data, bukan untuk video conferencing atau game online. Oleh karena itu, latensi 100 ms hingga 200 ms masih dapat diterima. Berdasarkan hasil pengujian kinerja *web dashboard*, dapat disimpulkan bahwa subobjektif 6, yaitu integrasi data yang akurat dan *real-time*, telah berhasil dipenuhi.

Hasil pengujian *backup* data menunjukkan bahwa data dari Antares berhasil di-backup dan disimpan pada database MySQL. Proses penyimpanan data ini dilakukan secara otomatis setiap kali user mengakses submenu dashboard. Dengan demikian, dapat disimpulkan bahwa subobjektif 7, yaitu penyediaan data backup, telah berhasil dipenuhi.

BAB V SIMPULAN DAN SARAN

Hasil pengujian menunjukkan bahwa subsistem dashboard yang dikembangkan berhasil memenuhi persyaratan desain yang ditetapkan. Pada desain *frontend*, dashboard mudah dinavigasi, informasi yang ditampilkan mudah dipahami, data dimuat dengan cepat, dan dashboard cocok untuk pemantauan sensor IoT, seperti yang dibuktikan oleh nilai indeks skala Likert yang tinggi. Hal ini menunjukkan bahwa subobjektif 8 tentang *user interface* yang mudah digunakan telah tercapai.

Pada desain *backend*, semua fitur berfungsi dengan baik dengan nilai latensi di bawah 200 ms, yang masih dapat diterima untuk aplikasi web pemantauan data. Latensi tertinggi terjadi pada submenu dashboard, disebabkan karena proses pengambilan dan pengecekan data dari database serta penampilan grafik data sensor IoT, namun tetap dalam batas yang dapat diterima. Proses *backup* data juga berhasil dilakukan, dengan memastikan data dari Antares disimpan dengan aman di database MySQL setiap kali submenu dashboard diakses. Dengan demikian, subobjektif 6 tentang integrasi data *real-time* dan subobjektif 7 tentang penyediaan data *backup* telah berhasil dipenuhi.

Berdasarkan penelitian tugas akhir yang telah dilakukan, terdapat beberapa saran yang untuk pengembangan lebih lanjut *web dashboard* pemantauan data sensor IoT:

1. Menambahkan fitur-fitur pada *web dashboard*, seperti panduan penggunaan *web dashboard*, tampilan lokasi (*maps*) dari device yang dimiliki user, pemilihan rentang waktu pada grafik yang lebih spesifik dengan menggunakan masukkan tanggal kalender, penyortiran data berdasarkan kolom tertentu, kustomisasi data yang ingin ditampilkan pada grafik atau tabel, dan pemberian peringatan jika ada data sensor yang memiliki nilai lebih atau kurang dari seharusnya.
2. Meningkatkan efisiensi proses pengambilan dan pengecekan data dari database untuk mengurangi latensi, terutama pada submenu dashboard dan data. Salah satu caranya adalah dengan menggunakan *indexing* pada kolom-

kolom yang sering digunakan dalam query serta menerapkan teknik *caching* untuk data yang sering diakses.

3. Mengimplementasikan load balancing untuk distribusi beban yang merata untuk meningkatkan kinerja *web dashboard* saat jumlah user meningkat.

DAFTAR PUSTAKA

- Abdullah, H. M., & Zeki, A. M. (2014). Frontend and backend web technologies in social networking sites: Facebook as an example. Proceedings - 3rd International Conference on Advanced Computer Science Applications and Technologies, ACSAT 2014, 85–89. <https://doi.org/10.1109/ACSAT.2014.22>
- Abinaya, T., Ishwarya, J., & Maheswari, M. (2019). A Novel Methodology for Monitoring and Controlling of Water Quality in Aquaculture using Internet of Things (IoT).
- Baddula, M., Ray, B., & Chowdhury, M. (2020, Desember 16). Performance Evaluation of Aloha and CSMA for LoRaWAN Network. 2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering, CSDE 2020. <https://doi.org/10.1109/CSDE50874.2020.9411539>
- Direktur Jendral Sumber Daya dan Perangkat Pos dan Informatika. (2019). Peraturan Direktur Jenderal Sumber Daya dan Perangkat Pos dan Informatika No. 3 Tahun 2019.
- FAO. (2024). The State of World Fisheries and Aquaculture 2024. Dalam The State of World Fisheries and Aquaculture 2024. FAO. <https://doi.org/10.4060/cd0683en>
- Guangyang, W., Na, X., Shun'an, X., & Yuhan, X. (2022). Development of Low Power Transmission Line Clamp Temperature Measurement System Based on Lora Communication. IEEE International Conference on Knowledge Engineering and Communication Systems, ICKES 2022. <https://doi.org/10.1109/ICKECS56523.2022.10060208>
- Gupta, A., Bansal, A., & Khanduja, V. (2017). Modern Lossless Compression Techniques: Review, Comparison and Analysis.
- Hassan, M. A. (2021). Relational and NoSQL databases: The appropriate database model choice. 2021 22nd International Arab Conference on Information Technology, ACIT 2021. <https://doi.org/10.1109/ACIT53391.2021.9677042>
- He, F. (2023). Implementation of Secure Login and Access Methods for Web Frontend. International Conference on Applied Intelligence and Sustainable

Computing, ICAISC 2023.

<https://doi.org/10.1109/ICAISC58445.2023.10199967>

Hidayati, V. N., Iskandar, & Satriobudi, A. B. (2022). Web Dashboard Development for Cloud Server-Based Air Quality Monitoring System. Proceeding of 2022 16th International Conference on Telecommunication Systems Services and Applications, TSSA 2022. <https://doi.org/10.1109/TSSA56819.2022.10063897>

Li, Y., & Manoharan, S. (2013). A performance comparison of SQL and NoSQL databases. IEEE Pacific Rim Conference On Communications, Computers and Signal Processing (PACRIM).

LZ4. (t.t.). LZ4. Diambil 29 Juli 2024, dari <https://lz4.org/>

Oberhummer. (2017). LZO. <https://www.oberhummer.com/opensource/lzo/>

Rani, M., & Singh, V. (2016). An Enhanced Text Compression System Based on ASCII Values and Huffman Coding. International Journal of Computer Science Trends and Technology (IJCS T), 4. www.ijcstjournal.org

Rivera Guzmán, E. F., Mañay Chochos, E. D., Chiliquinga Malliquinga, M. D., Baldeón Egas, P. F., & Toasa Guachi, R. M. (2022). LoRa Network-Based System for Monitoring the Agricultural Sector in Andean Areas: Case Study Ecuador. Sensors, 22(18). <https://doi.org/10.3390/s22186743>

siara-cc. (t.t.). Unishox2. Diambil 29 Juli 2024, dari <https://github.com/siara-cc/Unishox2>

Sukmandhani, A. A. (2020). QoS (Quality of Services). <https://online.binus.ac.id/computer-science/2020/06/15/qos-quality-of-services/>

Swanson, M., Bowen, P., Phillips, A. W., Gallup, D., & Lynes, D. (2010). Contingency planning guide for federal information systems. <https://doi.org/10.6028/NIST.SP.800-34r1>

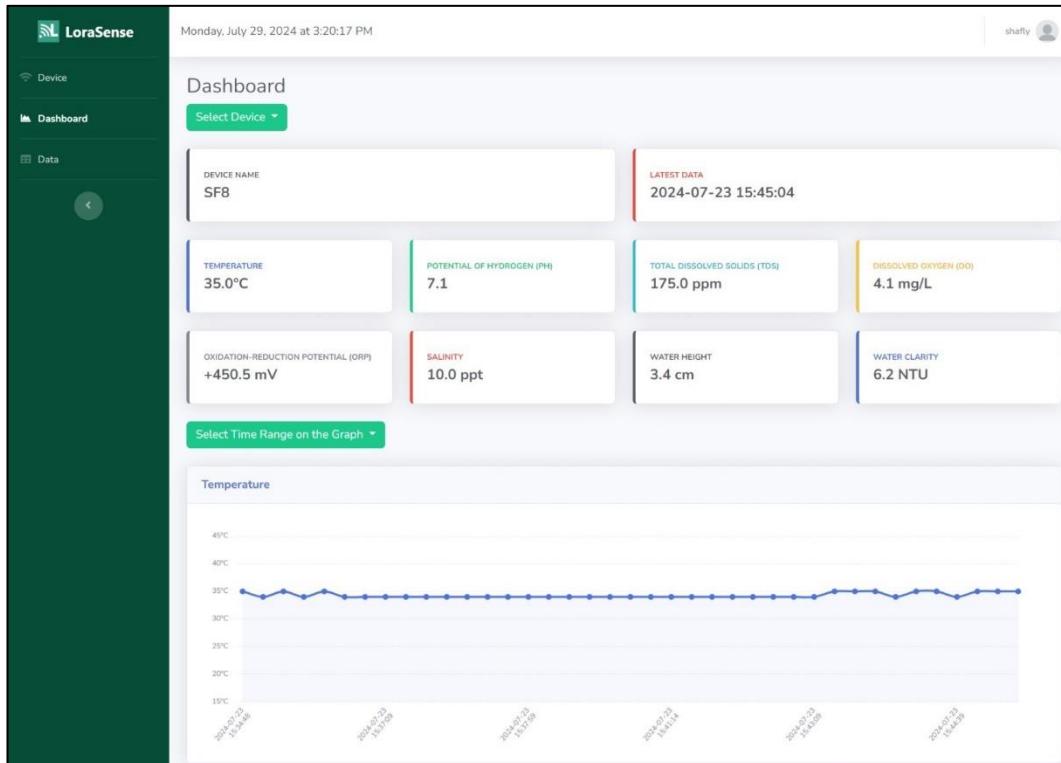
The Things Network. (t.t.-a). LoRaWAN Architecture. Diambil 29 Juli 2024, dari <https://www.thethingsnetwork.org/docs/lorawan/architecture/>

The Things Network. (t.t.-b). What are LoRa and LoRaWAN? Diambil 29 Juli 2024, dari <https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/>

Wahyuni Sabran, F., & Zalfiana Rusfian, E. (2023). Penggunaan Internet of Things pada eFishery untuk keberlanjutan Akuakultur di Indonesia. INNOVATIVE: Journal Of Social Science Research, 3, 8142–8156.

LAMPIRAN

LAMPIRAN A TAMPILAN WEB DASHBOARD



Submenu Dashboard

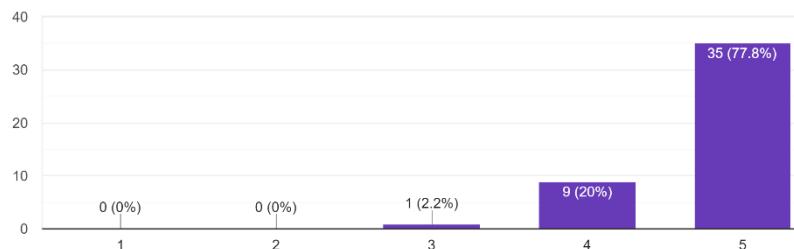
The screenshot shows the LoraSense web dashboard interface under the "Data" submenu. At the top, it displays the date and time as "Wednesday, July 24, 2024 at 8:37:23 PM" and the user "shafly". The sidebar menu shows "Device", "Dashboard", and "Data" (selected). The main content area is titled "Data" and features a "Select Device" dropdown set to "SF8". It shows a table titled "Data Table SF8" with the following columns: Device ID, Time, Temperature, pH, TDS, DO, ORP, Salinity, Water Height, and Water Clarity. The table contains 10 rows of data, all corresponding to the device ID 035303d90dcBee94, with timestamp intervals of approximately 1 minute. The data values are consistent with the real-time data shown in the previous screenshot.

Device ID	Time	Temperature	pH	TDS	DO	ORP	Salinity	Water Height	Water Clarity
035303d90dcBee94	2024-07-23 15:45:04	35.0°C	7.1	175.0 ppm	4.1 mg/L	450.5 mV	10.0 ppt	3.4 cm	6.2 NTU
035303d90dcBee94	2024-07-23 15:45:09	35.0°C	7.1	175.0 ppm	4.1 mg/L	450.5 mV	10.0 ppt	3.4 cm	6.2 NTU
035303d90dcBee94	2024-07-23 15:44:44	35.0°C	7.1	175.0 ppm	4.1 mg/L	450.5 mV	10.0 ppt	3.4 cm	6.2 NTU
035303d90dcBee94	2024-07-23 15:44:39	34.0°C	7.1	175.0 ppm	4.1 mg/L	450.5 mV	10.0 ppt	3.4 cm	6.2 NTU
035303d90dcBee94	2024-07-23 15:44:19	35.0°C	7.1	175.0 ppm	4.1 mg/L	450.5 mV	10.0 ppt	3.4 cm	6.2 NTU
035303d90dcBee94	2024-07-23 15:44:14	35.0°C	7.1	175.0 ppm	4.1 mg/L	450.5 mV	10.0 ppt	3.4 cm	6.2 NTU
035303d90dcBee94	2024-07-23 15:44:04	34.0°C	7.1	175.0 ppm	4.1 mg/L	450.5 mV	10.0 ppt	3.4 cm	6.2 NTU
035303d90dcBee94	2024-07-23 15:43:54	35.0°C	7.1	175.0 ppm	4.1 mg/L	450.5 mV	10.0 ppt	3.4 cm	6.2 NTU
035303d90dcBee94	2024-07-23 15:43:34	35.0°C	7.1	175.0 ppm	4.1 mg/L	450.5 mV	10.0 ppt	3.4 cm	6.2 NTU
035303d90dcBee94	2024-07-23 15:43:29	35.0°C	7.1	175.0 ppm	4.1 mg/L	450.5 mV	10.0 ppt	3.4 cm	6.2 NTU

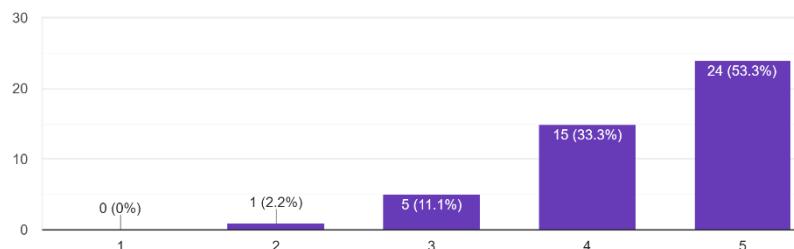
Submenu Data

LAMPIRAN B HASIL KUESIONER PENGUJIAN FRONTEND

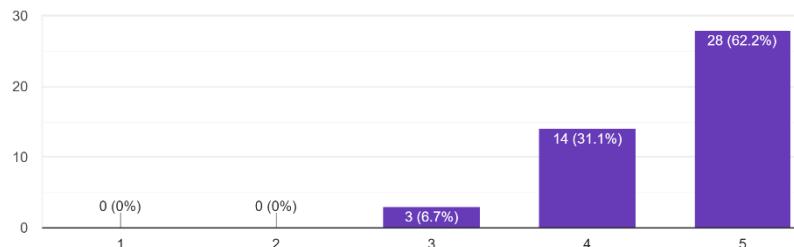
Seberapa mudah navigasi pada dashboard ini?
45 responses



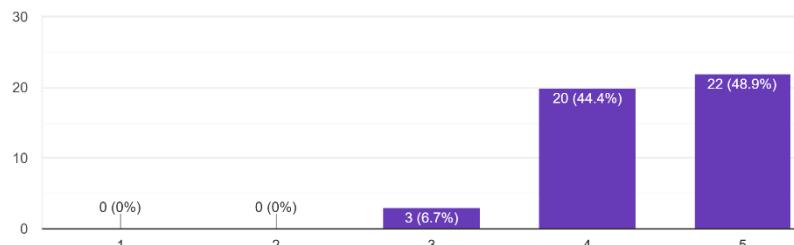
Apakah informasi dari data yang ditampilkan mudah dipahami?
45 responses



Seberapa cepat dashboard memuat data?
45 responses



Seberapa cocok dashboard tersebut digunakan untuk pemantauan data sensor IoT?
45 responses



LAMPIRAN C KODE PROGRAM FRONTEND

```

terminal Help ← → ⚡ Tugas Akhir
app.py database.py fetch_data.py Release Notes: 1.92.0 admin-users.html home.html admin-devices.html ...
templates > o home.html ...
1  {% extends "base.html" %}
2  {% block title %}Home{% endblock %}
3
4  {% block topbar %}
5  <!-- Top Navigation Links -->
6  <div class="navbar-nav">
7    <a class="nav-link active" href="{{ url_for('home') }}>Home</a>
8    <a class="nav-link" href="{{ url_for('login') }}>Login</a>
9    <a class="nav-link" href="{{ url_for('register') }}>Register</a>
10   </div>
11   {% endblock %}
12
13  {% block content %}
14  <!-- Main Content Section -->
15  <div class="container">
16    <!-- Logo and Tagline -->
17    <div style="text-align: center; margin: 20px 0;">
18      
19      <p class="responsive-tagline">Your IoT Sensor Monitoring Companion</p>
20    </div>
21
22    <!-- Introduction Section -->
23    <div class="jumbotron" style="background-color: #4dc7a2; border-radius: 1rem; padding: 2rem;">
24      <h1>loraSense</h1>
25      <p class="lead">
26        loraSense is an innovative IoT (Internet of Things) monitoring solution designed
27        to provide comprehensive insights into various sensor data using LoRa (long range) technology.
28      </p>
29      <hr class="my-4" />
30      <p>
31        loraSense for Aquaculture is a state-of-the-art IoT solution tailored for precision monitoring
32        of aquaculture environments. Using LoRa technology, it provides real-time insights into vital parameters
33        like water quality, temperature, pH, and dissolved oxygen, empowering aquaculture practitioners to maintain
34        optimal conditions for fish health and growth. With remote accessibility and advanced analytics,
35        loraSense streamlines operations, enhances productivity, and promotes sustainability in aquaculture
36        practices.
37      </p>
38      <!-- Button to Dashboard -->
39      <a class="btn btn-primary btn-lg" href="{{ url_for('device') }}" role="button">Go To Dashboard!</a>
40    </div>
41  </div>
42  {% endblock %}

```

Ln 43, Col 1 Spaces: 4 UTF-8 CRLF HTML ⚡ Go Live ✅ Prettier

Menu Home

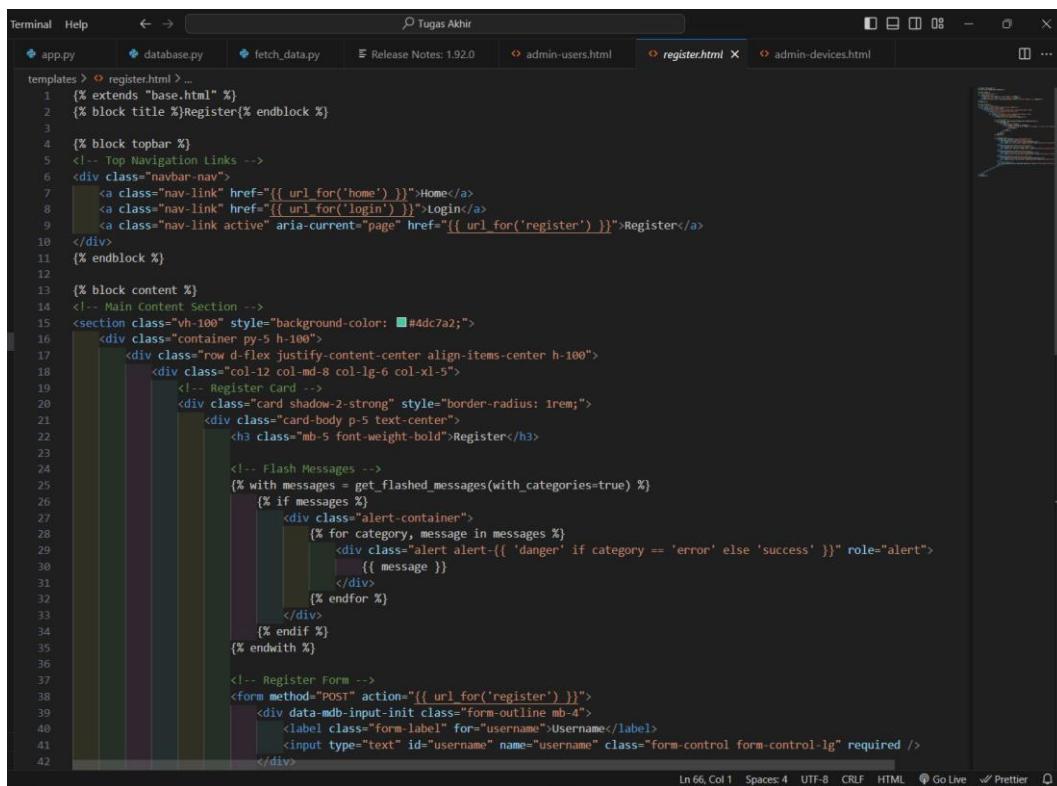
```

terminal Help ← → ⚡ Tugas Akhir
app.py database.py fetch_data.py Release Notes: 1.92.0 admin-users.html login.html admin-devices.html ...
templates > o login.html ...
1  {% extends "base.html" %}
2  {% block title %}Login{% endblock %}
3
4  {% block topbar %}
5  <!-- Top Navigation Links -->
6  <div class="navbar-nav">
7    <a class="nav-link" href="{{ url_for('home') }}>Home</a>
8    <a class="nav-link active" href="{{ url_for('login') }}>Login</a>
9    <a class="nav-link" href="{{ url_for('register') }}>Register</a>
10   </div>
11   {% endblock %}
12
13  {% block content %}
14  <!-- Main Content Section -->
15  <section class="vh-100" style="background-color: #4dc7a2;">
16    <div class="container py-5 h-100">
17      <div class="row d-flex justify-content-center align-items-center h-100">
18        <div class="col-12 col-md-8 col-lg-6 col-xl-5">
19          <!-- Login Card -->
20          <div class="card shadow-2-strong" style="border-radius: 1rem;">
21            <div class="card-body p-5 text-center">
22              <h3 class="mb-5 font-weight-bold">Login</h3>
23
24              <!-- Flash Messages -->
25              {% with messages = get_flashed_messages(with_categories=true) %}
26                {% if messages %}
27                  {% for message in messages %}
28                    <div class="alert alert-{{ 'danger' if category == 'error' else 'success' }} role="alert">
29                      {{ message }}
30                    </div>
31                  {% endfor %}
32                {% endif %}
33              {% endwith %}
34
35              <!-- Login Form -->
36              <form method="POST" action="{{ url_for('login') }}>
37                <div data-mdb-input-init class="form-outline mb-4">
38                  <label class="form-label" for="username">Username</label>
39                  <input type="text" id="username" name="username" class="form-control form-control-lg" required />
40                </div>
41
42

```

Ln 60, Col 1 Spaces: 4 UTF-8 CRLF HTML ⚡ Go Live ✅ Prettier

Menu Login



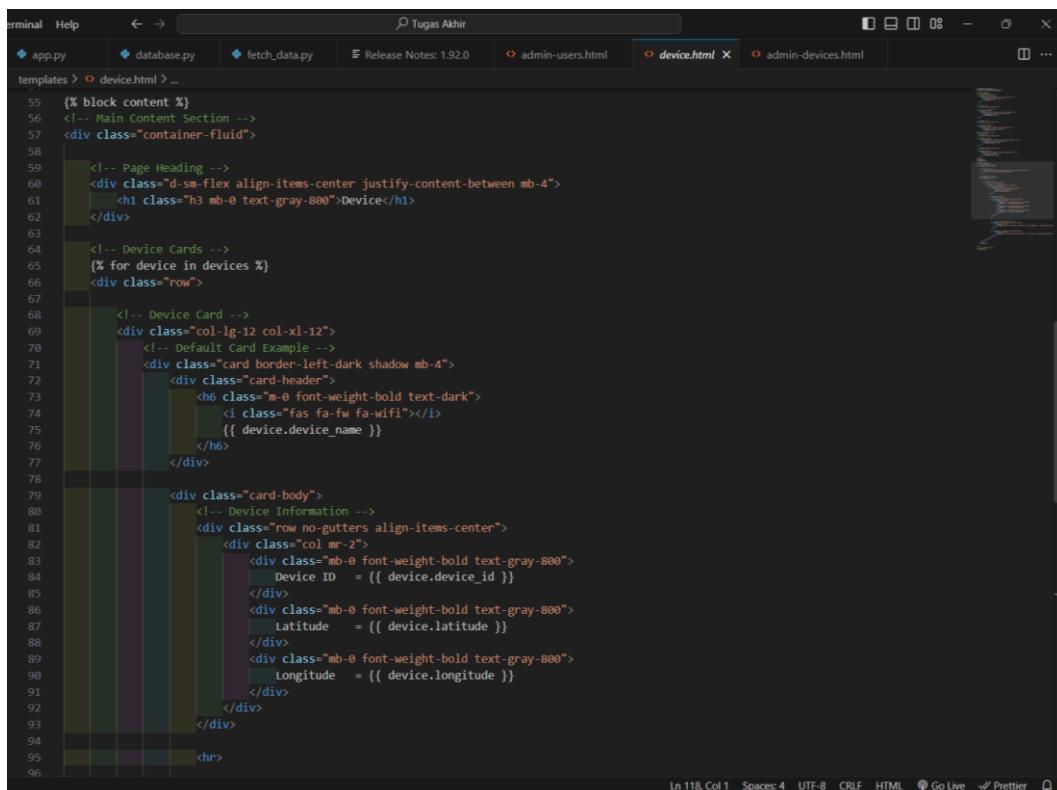
```

Terminal Help ← → Tugas Akhir
app.py database.py fetch_data.py Release Notes: 1.92.0 register.html admin-users.html admin-devices.html
templates > register.html ...
1  {% extends "base.html" %}
2  {% block title %}Register{% endblock %}
3
4  {% block topbar %}
5      <!-- Top Navigation Links -->
6      <div class="navbar-nav">
7          <a class="nav-link" href="{{ url_for('home') }}>Home</a>
8          <a class="nav-link" href="{{ url_for('login') }}>Login</a>
9          <a class="nav-link active" aria-current="page" href="{{ url_for('register') }}>Register</a>
10     </div>
11 {% endblock %}
12
13 {% block content %}
14     <!-- Main Content Section -->
15     <section class="vh-100" style="background-color: #4dc7a2;">
16         <div class="container py-5 h-100">
17             <div class="row d-flex justify-content-center align-items-center h-100">
18                 <div class="col-12 col-md-8 col-lg-6 col-xl-5">
19                     <!-- Register Card -->
20                     <div class="card shadow-2-strong" style="border-radius: 1rem;">
21                         <div class="card-body p-5 text-center">
22                             <h3 class="mb-5 font-weight-bold">Register</h3>
23
24                             <!-- Flash Messages -->
25                             {% with messages = get_flashed_messages(with_categories=true) %}
26                                 {% if messages %}
27                                     <div class="alert-container">
28                                         {% for category, message in messages %}
29                                             <div class="alert alert-{{ category }} role="alert">
30                                                 {{ message }}
31                                             </div>
32                                         {% endif %}
33                                     {% endif %}
34                                 {% endwith %}
35
36                             <!-- Register Form -->
37                             <form method="POST" action="{{ url_for('register') }}>
38                                 <div data-mdb-input-init class="form-outline mb-4">
39                                     <label class="form-label" for="username">Username</label>
40                                     <input type="text" id="username" name="username" class="form-control form-control-lg" required />
41                             </div>
42

```

Ln 66, Col 1 Spaces: 4 UTF-8 CRLF HTML Go Live Prettier

Menu Register



```

Terminal Help ← → Tugas Akhir
app.py database.py fetch_data.py Release Notes: 1.92.0 device.html admin-users.html admin-devices.html
templates > device.html ...
55  {% block content %}
56  <!-- Main Content Section -->
57  <div class="container-fluid">
58
59      <!-- Page Heading -->
60      <div class="d-sm-flex align-items-center justify-content-between mb-4">
61          <h1 class="h3 mb-0 text-gray-800">Device</h1>
62      </div>
63
64      <!-- Device Cards -->
65      {% for device in devices %}
66          <div class="row">
67
68              <!-- Device Card -->
69              <div class="col-lg-12 col-xl-12">
70                  <!-- Default Card Example -->
71                  <div class="card border-left-dark shadow mb-4">
72                      <div class="card-header">
73                          <h6 class="m-0 font-weight-bold text-dark">
74                              <i class="fas fa-fw fa-wifi"></i>
75                              {{ device.device_name }}
76                          </h6>
77                      </div>
78
79                      <div class="card-body">
80                          <!-- Device Information -->
81                          <div class="row no-gutters align-items-center">
82                              <div class="col mr-2">
83                                  <div class="mb-0 font-weight-bold text-gray-800">
84                                      Device ID = {{ device.device_id }}
85                                  </div>
86                                  <div class="mb-0 font-weight-bold text-gray-800">
87                                      Latitude = {{ device.latitude }}
88                                  </div>
89                                  <div class="mb-0 font-weight-bold text-gray-800">
90                                      Longitude = {{ device.longitude }}
91                                  </div>
92                          </div>
93                      </div>
94
95                  <hr>

```

Ln 118, Col 1 Spaces: 4 UTF-8 CRLF HTML Go Live Prettier

Submenu Device

```

56  {% block content %}
57  <!-- Main Content Section -->
58  <div class="container-fluid">
59
60    <!-- Page Heading -->
61    <div class="d-sm-flex align-items-center justify-content-between mb-2">
62      <h1 class="h3 mb-0 text-gray-800">Dashboard</h1>
63    </div>
64
65    <!-- Device Selection Dropdown -->
66    <div class="dropdown mb-4">
67      <button class="btn btn-success dropdown-toggle" type="button" id="dropdownMenuButton" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">Select Device</button>
68      <div class="dropdown-menu" aria-labelledby="dropdownMenuButton">
69        <!-- Device Dropdown Items -->
70        {% for device in devices %}
71          <a class="dropdown-item" href="{{ url_for('dashboard', device_id=device.device_id, time_range=time_range) }}>{{ device.device_name }}</a>
72        {% endfor %}
73      </div>
74    </div>
75
76    <!-- Content Rows -->
77    <div class="row">
78
79      <!-- Device Name Card -->
80      <div class="col-1x-6 col-lg-6 col-md-12 mb-4">
81        <!-- Device Name Display -->
82        <div class="card border-left-dark shadow h-100 py-2">
83          <div class="card-body">
84            <div class="row no-gutters align-items-center">
85              <div class="col mr-2">
86                <div class="text-xs font-weight-bold text-dark text-uppercase mb-1">
87                  Device Name
88                </div>
89                <div class="h5 mb-0 font-weight-bold text-gray-800">
90                  {{ device.device_name | default('No Device Selected') }}
91                </div>
92              </div>
93            </div>
94          </div>
95        </div>
96      </div>
97    </div>

```

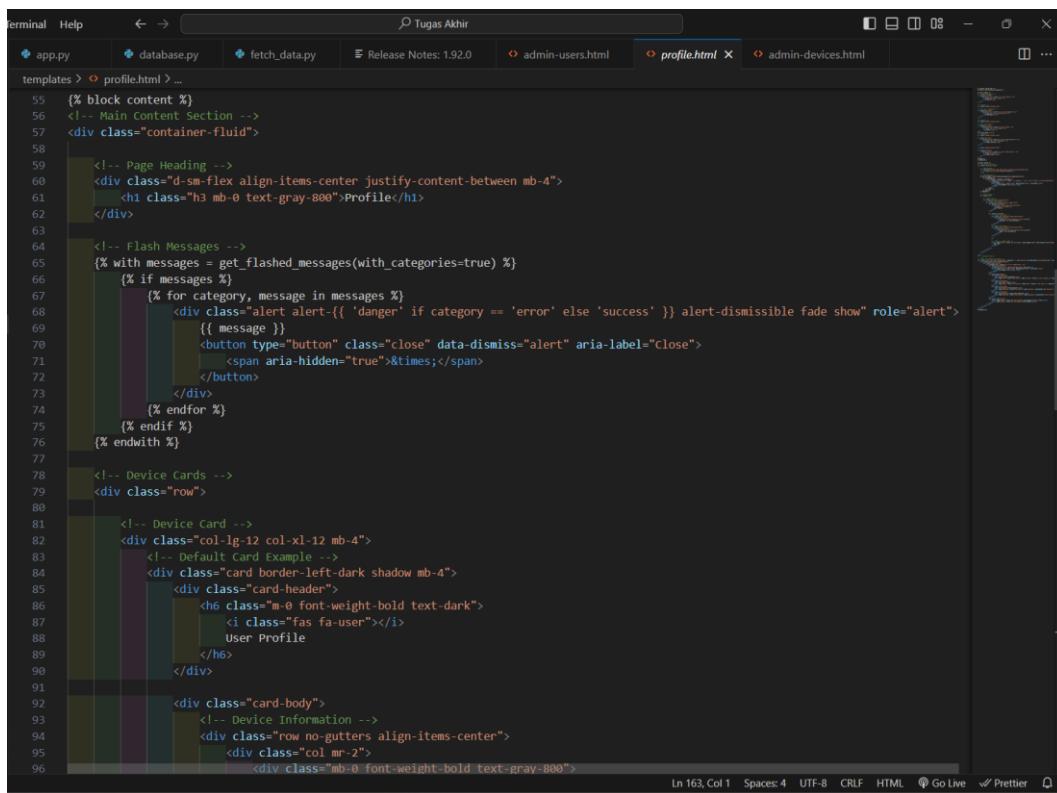
Submenu Dashboard

```

47  {% block content %}
48  <div class="container-fluid">
49    <div class="d-sm-flex align-items-center justify-content-between mb-2">
50      <h1 class="h3 mb-0 text-gray-800">Data</h1>
51    </div>
52
53    {% with messages = get_flashed_messages(with_categories=true) %}
54      {% if messages %}
55        {% for category, message in messages %}
56          <div class="alert alert-{{ 'danger' if category == 'error' else 'success' }} alert-dismissible fade show" role="alert">
57            {{ message }}
58            <button type="button" class="close" data-dismiss="alert" aria-label="Close">&times;</button>
59          </div>
60        {% endif %}
61      {% endif %}
62
63    <!-- Device Selection Dropdown -->
64    <div class="dropdown mb-4">
65      <button class="btn btn-success dropdown-toggle" type="button" id="dropdownMenuButton" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">Select Device</button>
66      <div class="dropdown-menu" aria-labelledby="dropdownMenuButton">
67        {% for device in devices %}
68          <a class="dropdown-item" href="{{ url_for('data', device_id=device.device_id) }}>{{ device.device_name }}</a>
69        {% endfor %}
70      </div>
71    </div>
72
73    <div class="card shadow mb-4">
74      <div class="card-header py-3">
75        <div class="d-sm-flex align-items-center justify-content-between">
76          <h6 class="m-0 font-weight-bold text-dark">
77            Data Table {{ device.device_name | default('No Device Selected') }}
78          </h6>
79          <form id="download-form" action="{{ url_for('download_csv') }}" method="post">
80            <input type="hidden" name="selected_device_id" value="{{ device.device_id }}>
81            <button type="submit" class="btn btn-sm btn-success">
82              <i class="fas fa-download fa-sm text-white-50"></i> Download Data
83            </button>
84          </form>
85        </div>
86      </div>
87    </div>

```

Submenu Data



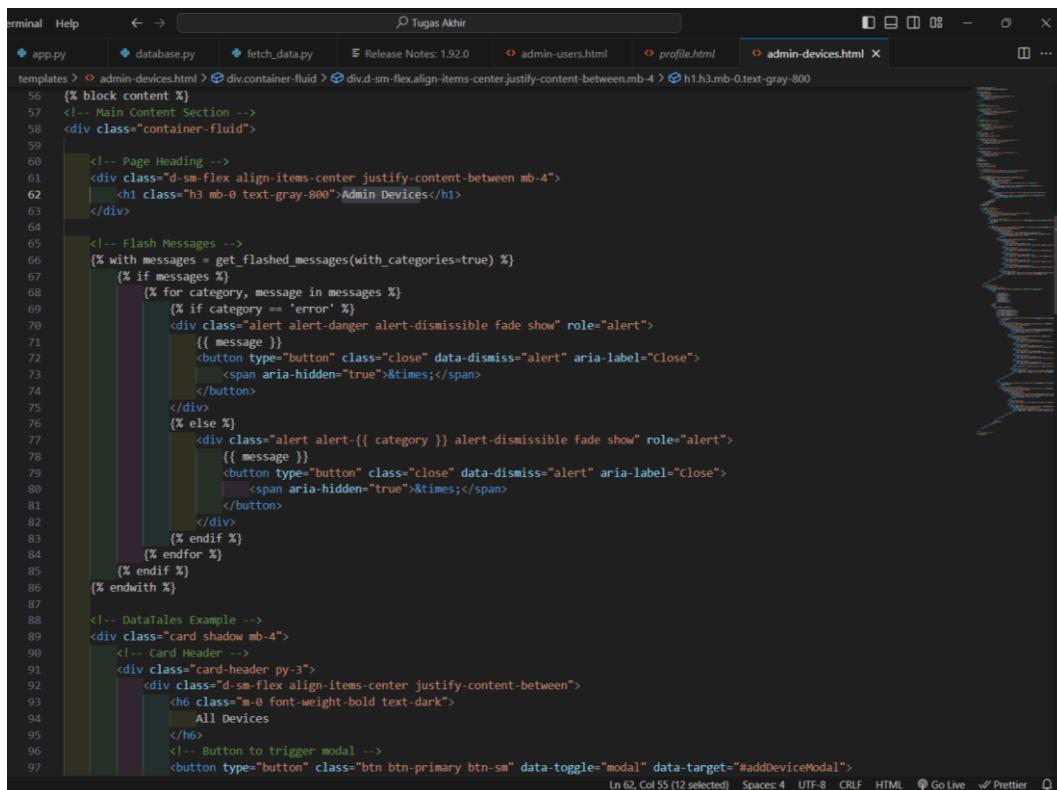
```

55  {% block content %}
56  <!-- Main Content Section -->
57  <div class="container-fluid">
58
59  <!-- Page Heading -->
60  <div class="d-sm-flex align-items-center justify-content-between mb-4">
61  |   <h1 class="h3 mb-0 text-gray-800">Profile</h1>
62  </div>
63
64  <!-- Flash Messages -->
65  {% with messages = get_flashed_messages(with_categories=true) %}
66  |   {% if messages %}
67  |       {% for category, message in messages %}
68  |           <div class="alert alert-{{ 'danger' if category == 'error' else 'success' }} alert-dismissible fade show" role="alert">
69  |               {{ message }}
70  |               <button type="button" class="close" data-dismiss="alert" aria-label="Close">
71  |                   <span aria-hidden="true">&times;</span>
72  |               </button>
73  |           </div>
74  |       {% endif %}
75  |   {% endif %}
76  |   {% endwith %}
77
78  <!-- Device Cards -->
79  <div class="row">
80
81      <!-- Device Card -->
82      <div class="col-lg-12 col-xl-12 mb-4">
83          <!-- Default Card Example -->
84          <div class="card border-left-dark shadow mb-4">
85              <div class="card-header">
86                  <h6 class="m-0 font-weight-bold text-dark">
87                      <i class="fas fa-user"></i>
88                      User Profile
89                  </h6>
90              </div>
91
92              <div class="card-body">
93                  <!-- Device Information -->
94                  <div class="row no-gutters align-items-center">
95                      <div class="col mr-2">
96                          <div class="mb-0 font-weight-bold text-gray-800">

```

Ln 163, Col 1 Spaces: 4 UTF-8 CRLF HTML ⚡ Go Live ✨ Prettier

Submenu Profile



```

56  {% block content %}
57  <!-- Main Content Section -->
58  <div class="container-fluid">
59
60  <!-- Page Heading -->
61  <div class="d-sm-flex align-items-center justify-content-between mb-4">
62  |   <h1 class="h3 mb-0 text-gray-800">Admin Devices</h1>
63  </div>
64
65  <!-- Flash Messages -->
66  {% with messages = get_flashed_messages(with_categories=true) %}
67  |   {% if messages %}
68  |       {% for category, message in messages %}
69  |           <div class="alert alert-{{ 'danger' if category == 'error' else 'success' }} alert-dismissible fade show" role="alert">
70  |               {{ message }}
71  |               <button type="button" class="close" data-dismiss="alert" aria-label="Close">
72  |                   <span aria-hidden="true">&times;</span>
73  |               </button>
74  |           </div>
75  |       {% endif %}
76  |   {% endif %}
77  |   {% endwith %}
78
79  <!-- DataTales Example -->
80  <div class="card shadow mb-4">
81      <!-- Card Header -->
82      <div class="card-header py-3">
83          <div class="d-sm-flex align-items-center justify-content-between">
84              <h6 class="m-0 font-weight-bold text-dark">
85                  All Devices
86              </h6>
87              <!-- Button to trigger modal -->
88              <button type="button" class="btn btn-primary btn-sm" data-toggle="modal" data-target="#addDeviceModal">

```

Ln 62, Col 55 (12 selected) Spaces: 4 UTF-8 CRLF HTML ⚡ Go Live ✨ Prettier

Submenu Admin Devices

```

56  {% block content %} ...
57  <!-- Main Content Section -->
58  <div class="container-fluid">
59
60    <!-- Page Heading -->
61    <div class="d-sm-flex align-items-center justify-content-between mb-4">
62      <h1 class="h3 mb-0 text-gray-800">Admin Users</h1>
63    </div>
64
65    <!-- DataTales Example -->
66    <div class="card shadow mb-4">
67      <!-- Card Header -->
68      <div class="card-header py-3">
69        <h6 class="m-0 font-weight-bold text-dark">
70          All Users
71        </h6>
72      </div>
73      <!-- Card Body -->
74      <div class="card-body">
75        <div class="table-responsive">
76          <table class="table table-bordered" id="dataTable" width="100%" cellspacing="0">
77            <thead>
78              <tr>
79                <th>Username</th>
80                <th>User ID</th>
81                <th>Email</th>
82                <th>Is Admin</th>
83              </tr>
84            </thead>
85            <tbody>
86              {# for user in users %}
87              <tr>
88                <td>{{ user.username }}</td>
89                <td>{{ user.user_id }}</td>
90                <td>{{ user.email }}</td>
91                <td>{{ user.admin }}</td>
92              </tr>
93              {# endfor %}

```

Submenu Admin Users

```

1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5    <meta charset="UTF-8">
6    <meta name="viewport" content="width=device-width, initial-scale=1.0">
7    <title>
8      {# block title #}{% endblock %}
9    </title>
10   <link rel="icon" href="{{ url_for('static', filename='img/logo.icon.ico') }}" type="image/x-icon">
11   <link href="{{ url_for('static', filename='css/lorasense.css') }}" rel="stylesheet">
12 </head>
13
14 <body>
15   <!-- Navbar -->
16   <nav class="navbar navbar-expand-lg navbar-dark bg-primary fixed-top">
17     <div class="container-fluid">
18       <a class="navbar-brand" href="{{ url_for('home') }}>
19         
20         LoraSense
21     </div>

```

base.html

```

1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5    <meta charset="utf-8">
6    <meta http-equiv="X-UA-Compatible" content="IE=edge">
7    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
8    <title>
9      {# block title #}{% endblock %}
10     </title>
11     <link rel="icon" href="{{ url_for('static', filename='img/logo.icon.ico') }}" type="image/x-icon">
12
13     <!-- Custom fonts -->
14     <link href="{{ url_for('static', filename='vendor/fontawesome-free/css/all.min.css') }}" rel="stylesheet" type="text/css">
15     <link href="https://fonts.googleapis.com/css?family=Nunito:200,200i,300,300i,400,400i,600,600i,700,700i,800,800i,900,900i" rel="stylesheet">
16
17     <!-- Custom styles for this template -->
18     <link href="{{ url_for('static', filename='css/lorasense.css') }}" rel="stylesheet">
19
20   </head>
21
22 <body id="page-top">

```

base2.html

LAMPIRAN D KODE PROGRAM BACKEND FLASK APP

```
"""
app.py

This module contains the Flask application for managing a system that handles
user registration, authentication, device management, and data retrieval. The
application provides endpoints for users to register, log in, add and manage
devices, view and analyze data collected from these devices, and other related
functionalities. It also includes error handling, session management, and data
validation to ensure secure and efficient operation.
"""

import os
import shutil
import tempfile
import csv
from datetime import datetime, timedelta
from functools import wraps
import jwt
from flask import (
    Flask, g, request, render_template, redirect, url_for,
    make_response, send_file, flash
)
from database import (
    add_device_to_db, add_user, delete_device_by_id, get_all_devices, device_exists,
    get_all_users, get_user_id, get_data_by_device_and_time, get_device_data,
    get_device_info, get_device_last_data, get_user_devices, get_user_info,
    is_admin, login_user, store_data, update_device_in_db, update_user_info,
    user_exists
)

app = Flask(__name__)

# Load configuration from environment or default
app.config['SECRET_KEY'] = os.getenv('SECRET_KEY', 'default_secret_key')

# Token generation and verification
def generate_token(user_id):
    """Generate a JWT token with the user ID"""
    expiration = datetime.utcnow() + timedelta(minutes=30)
    return jwt.encode(
        {'user_id': user_id, 'exp': expiration}, app.config['SECRET_KEY'],
        algorithm='HS256'
    )

def verify_token(token):
    """Verify a JWT token and decode it"""
    try:
        return jwt.decode(token, app.config['SECRET_KEY'], algorithms=['HS256'])
    except jwt.ExpiredSignatureError:
        return None # Token has expired
    except jwt.InvalidTokenError:
        return None # Token is invalid

def token_required(f):
    """Decorator that ensures a valid token is present for the endpoint"""
    @wraps(f)
    def decorated(*args, **kwargs):
        token = request.cookies.get('token')
        if not token or not (decoded_token := verify_token(token)):
            return redirect(url_for('login'))
        g.current_user = decoded_token['user_id']
        return f(*args, **kwargs)
    return decorated

def admin_required(f):
    """Decorator that ensures the current user is an admin"""
    @wraps(f)
    def decorated(*args, **kwargs):
```

```

        token = request.cookies.get('token')
        if not token or not (decoded_token := verify_token(token)):
            return redirect(url_for('login'))
        g.current_user = decoded_token['user_id']
        user_id = g.current_user
        if not is_admin(user_id):
            return redirect(request.referrer)
        return f(*args, **kwargs)
    return decorated

# Error handler
@app.errorhandler(Exception)
def handle_exception(error):
    """Global error handler for handling exceptions"""
    app.logger.error(f"An error occurred: {error}")
    if request.referrer:
        return redirect(request.referrer)
    return redirect(url_for('home'))

# Routes
@app.route('/')
def home():
    """Render the home page"""
    return render_template('home.html')

@app.route('/register', methods=['GET', 'POST'])
def register():
    """Handle user registration"""
    if request.method == 'POST':
        username = request.form.get('username')
        email = request.form.get('email')
        password = request.form.get('password')
        password2 = request.form.get('password2')

        if password != password2:
            flash('Passwords do not match.', 'error')
            return redirect(url_for('register'))

        if user_exists(username) or user_exists(email):
            flash('Username or Email already exists.', 'error')
            return redirect(url_for('register'))

        success = add_user({
            'username': username,
            'email': email,
            'password': password
        })

        if success:
            flash('User registered successfully.', 'success')
            return redirect(url_for('login'))

        flash('Failed to register user.', 'error')
        return redirect(url_for('register'))

    return render_template('register.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    """Handle user login"""
    if request.method == 'POST':
        auth = request.form

        if login_user(auth):
            username = request.form.get('username')
            user_id = get_user_id(username)
            token = generate_token(user_id)
            redirect_page = 'device'
            response = make_response(redirect(url_for(redirect_page)))
            response.set_cookie('token', token, httponly=True)

```

```

        return response

    flash('Invalid username or password.', 'error')
    return redirect(url_for('login'))

    return render_template('login.html')

@app.route('/logout')
def logout():
    """Handle user logout"""
    response = make_response(redirect(url_for('login')))
    response.set_cookie('token', '', expires=0)
    return response

@app.route('/device')
@token_required
def device():
    """Render the user's devices page"""
    try:
        current_user_id = g.current_user
        user_info = get_user_info(current_user_id)
        user_devices = get_user_devices(current_user_id)
        return render_template('device.html', devices=user_devices, user=user_info)
    except Exception as e:
        app.logger.error(f"Error occurred: {e}")
        return redirect(request.referrer)

@app.route('/dashboard', methods=['GET', 'POST'])
@token_required
def dashboard():
    """Render the device dashboard page"""
    try:
        current_user_id = g.current_user
        user_info = get_user_info(current_user_id)
        user_devices = get_user_devices(current_user_id)
        selected_device_id = request.args.get('device_id')
        selected_time_range = request.args.get('time_range')

        selected_device_info = selected_device_last_data = None
        selected_device_data = []

        if selected_device_id and selected_device_id not in [
            device.device_id for device in user_devices]:
            return render_template('dashboard.html', time_range=selected_time_range,
                                  latest=selected_device_last_data, device_data=selected_device_data,
                                  device=selected_device_info, devices=user_devices, user=user_info)

        if selected_device_id:
            store_data(selected_device_id)

            selected_device_info = get_device_info(selected_device_id)
            selected_device_last_data = get_device_last_data(selected_device_id)
            selected_device_data = get_data_by_device_and_time(selected_device_id,
            selected_time_range)

            return render_template('dashboard.html', time_range=selected_time_range,
                                  latest=selected_device_last_data, device_data=selected_device_data,
                                  device=selected_device_info, devices=user_devices, user=user_info)
        except Exception as e:
            app.logger.error(f"Error occurred: {e}")
            return redirect(request.referrer)

@app.route('/data', methods=['GET', 'POST'])
@token_required
def data():
    """Render the device data page"""
    try:
        current_user_id = g.current_user
        user_info = get_user_info(current_user_id)
        user_devices = get_user_devices(current_user_id)

```

```

selected_device_id = request.args.get('device_id')
selected_device_info = None
selected_device_data = []

if selected_device_id and selected_device_id not in [
    device.device_id for device in user_devices]:
    return render_template(
        'data.html',
        data=selected_device_data,
        device=selected_device_info,
        devices=user_devices,
        user=user_info
    )
if selected_device_id:
    store_data(selected_device_id)

selected_device_data = get_device_data(selected_device_id)
selected_device_info = get_device_info(selected_device_id)

return render_template('data.html', data=selected_device_data,
                      device=selected_device_info, devices=user_devices, user=user_info)
except Exception as e:
    app.logger.error(f"Error occurred: {e}")
    return redirect(request.referrer)

@app.route('/download_csv', methods=['POST'])
@token_required
def download_csv():
    """Download device data as a CSV file"""
    current_user_id = g.current_user
    selected_device_id = request.form.get('selected_device_id')
    user_devices = get_user_devices(current_user_id)

    # If no device is selected, redirect back with flash message
    if not selected_device_id:
        flash('Please select a device before downloading data.', 'error')
        return redirect(request.referrer)

    # Verify if the selected device belongs to the current user
    if selected_device_id not in [device.device_id for device in user_devices]:
        flash('Unauthorized.', 'error')
        return redirect(request.referrer)

    selected_device_data = get_device_data(selected_device_id)
    selected_device_info = get_device_info(selected_device_id)

    # Create CSV data
    csv_data = [[ "device_id", "time", "temp", "ph", "tds", "do", "orp", "salinity",
    "water_h", "water_cl"]]
    for row in selected_device_data:
        csv_data.append([
            row.device_id,
            str(row.time),
            str(row.temp),
            str(row.ph),
            str(row.tds),
            str(row.do),
            str(row.orp),
            str(row.salinity),
            str(row.water_h),
            str(row.water_cl)
        ])
    # Generate file name based on selected device
    csv_filename = f"{selected_device_info.device_name}_data.csv"

    # Create temporary directory
    temp_dir = tempfile.mkdtemp()

```

```

# Create CSV file in temporary directory
csv_filepath = os.path.join(temp_dir, csv_filename)
with open(csv_filepath, 'w', newline='') as csv_file:
    writer = csv.writer(csv_file)
    writer.writerows(csv_data)

# Create response and remove temp_dir after sending file
try:
    response = send_file(csv_filepath, as_attachment=True)
    response.call_on_close(lambda: shutil.rmtree(temp_dir))
    return response
except Exception as e:
    app.logger.error(f"Error occurred while sending file: {e}")
    shutil.rmtree(temp_dir)
    flash('Failed to download data.', 'error')
    return redirect(request.referrer)

@app.route('/admin-devices')
@admin_required
def admin_devices():
    """Render the admin devices management page"""
    try:
        current_user_id = g.current_user
        user_info = get_user_info(current_user_id)
        all_devices = get_all_devices()
        all_users = get_all_users()
        merged_data = []
        for device in all_devices:
            for user in all_users:
                if device.user_id == user.user_id:
                    merged_data.append({
                        'username': user.username,
                        'device_name': device.device_name,
                        'device_id': device.device_id,
                        'latitude': device.latitude,
                        'longitude': device.longitude
                    })
        return render_template('admin-devices.html', devices=merged_data,
user=user_info)
    except Exception as e:
        app.logger.error(f"Error occurred: {e}")
        return redirect(request.referrer)

@app.route('/add_device', methods=['POST'])
@admin_required
def add_device():
    """Add a new device to the database"""
    if request.method == 'POST':
        data = request.form
        username = data.get('username')
        device_id = data.get('device_id')
        device_name = data.get('device_name')
        latitude = data.get('latitude')
        longitude = data.get('longitude')
        # Validation
        if user_exists(username):
            user_id = get_user_id(username)
            if device_exists(device_id) or device_exists(device_name):
                flash('Device ID or Device Name already exists.', 'error')
            else:
                success = add_device_to_db(user_id, device_id, device_name, latitude,
longitude)
                if success:
                    flash('Device added successfully.', 'success')
                else:
                    flash('Failed to add device.', 'error')
        else:
            flash('Username does not exist.', 'error')
    return redirect(url_for('admin_devices'))

```

```

@app.route('/update_device', methods=['POST'])
@admin_required
def update_device():
    """Update device information in the database"""
    if request.method == 'POST':
        data = request.form
        username = data.get('username')
        old_device_id = data.get('old_device_id')
        device_name = data.get('device_name')
        latitude = data.get('latitude')
        longitude = data.get('longitude')
        device_info = get_device_info(old_device_id)
        # Validation
        if user_exists(username):
            user_id = get_user_id(username)
            if device_name != device_info.device_name and device_exists(device_name):
                flash('Device Name already exists.', 'error')
            else:
                success = update_device_in_db(old_device_id, user_id, device_name,
                                                latitude, longitude)
                if success:
                    flash('Device information updated successfully.', 'success')
                else:
                    flash('Failed to update device information.', 'error')
        else:
            flash('Username does not exist.', 'error')
        return redirect(url_for('admin_devices'))

@app.route('/delete_device', methods=['POST'])
@admin_required
def delete_device():
    """Delete a device from the database"""
    if request.method == 'POST':
        device_id = request.form.get('device_id')
        if device_id:
            success = delete_device_by_id(device_id)
            if success:
                flash('Device deleted successfully.', 'success')
            else:
                flash('Failed to delete device.', 'error')
        else:
            flash('Device ID not provided.', 'error')
        return redirect(url_for('admin_devices'))

@app.route('/admin-users')
@admin_required
def admin_users():
    """Render the admin users management page"""
    try:
        current_user_id = g.current_user
        user_info = get_user_info(current_user_id)
        all_users = get_all_users()
        return render_template('admin-users.html', users=all_users, user=user_info)
    except Exception as e:
        app.logger.error(f"Error occurred: {e}")
        return redirect(request.referrer)

@app.route('/profile')
@token_required
def profile():
    """Render user's profile page"""
    try:
        current_user_id = g.current_user
        user_info = get_user_info(current_user_id)
        return render_template('profile.html', user=user_info)
    except Exception as e:
        app.logger.error(f"Error occurred: {e}")
        return redirect(request.referrer)

@app.route('/update_user', methods=['POST'])

```

```

@token_required
def update_user():
    """Update user profile information"""
    try:
        current_user_id = g.current_user
        user_info = get_user_info(current_user_id)
        new_username = request.form.get('username')
        new_email = request.form.get('email')
        new_password = request.form.get('password')
        new_password2 = request.form.get('password2')

        if new_password != new_password2:
            flash('Passwords do not match.', 'error')
        elif new_username != user_info.username and user_exists(new_username):
            flash('Username already exists.', 'error')
        else:
            success = update_user_info(current_user_id, new_username, new_email,
new_password)
            if success:
                flash('Profile updated successfully.', 'success')
            else:
                flash('Failed to update profile.', 'error')
        return redirect(url_for('profile'))
    except Exception as e:
        app.logger.error(f"Error occurred: {e}")
        return redirect(request.referrer)

def main():
    """Entry point for running the Flask application"""
    app.run(debug=True)

if __name__ == '__main__':
    main()

```

LAMPIRAN E KODE PROGRAM BACKEND DATABASE

```
"""
database.py

This module contains SQLAlchemy ORM definitions and database operations for a
system managing users, devices, and data entries. It includes classes for User,
Device, and Data, along with functions to interact with these entities such as
adding users, retrieving device data, and storing data fetched from an external API.

"""

import os
import json
from datetime import datetime, timedelta
import bcrypt
from sqlalchemy import (
    create_engine, desc, Column, Float, DateTime,
    Boolean, String, ForeignKey, Integer, or_
)
from sqlalchemy.orm import declarative_base, sessionmaker, scoped_session, relationship
from sqlalchemy.exc import IntegrityError
from dotenv import load_dotenv
from fetch_data import fetch_data_api_antares

# Load environment variables from the .env file
load_dotenv()

# Retrieve the variables from the environment
username = os.getenv('DB_USERNAME')
password = os.getenv('DB_PASSWORD')
hostname = os.getenv('DB_HOST')
port = os.getenv('DB_PORT')
database_name = os.getenv('DB_NAME')
app_lora = os.getenv('APP_LORA')
api_key = os.getenv('API_KEY_ANTARES')

# Construct the connection URL
connection_url =
f'mysql+pymysql://{{username}}:{{password}}@{{hostname}}:{{port}}/{{database_name}}'

# Create the SQLAlchemy engine
engine = create_engine(connection_url)

# Define the base class for declarative class definitions
Base = declarative_base()

class User(Base):
    """Represents a user in the database."""
    __tablename__ = 'user'
    user_id = Column(Integer, primary_key=True, autoincrement=True)
    username = Column(String(100))
    email = Column(String(100))
    password = Column(String(100))
    admin = Column(Boolean, default=False)

    devices = relationship('Device', back_populates='user', cascade='all, delete-orphan')

    def set_password(self, password):
        """Hashes and sets the user's password"""
        self.password = bcrypt.hashpw(password.encode('utf-8'),
        bcrypt.gensalt()).decode('utf-8')

    def check_password(self, password):
        """Checks if the provided password matches the user's hashed password"""
        return bcrypt.checkpw(password.encode('utf-8'), self.password.encode('utf-8'))

class Device(Base):
    """Represents a device in the database."""
    __tablename__ = 'device'
```

```

device_id = Column(String(100), primary_key=True)
device_name = Column(String(100))
latitude = Column(String(100))
longitude = Column(String(100))
user_id = Column(Integer, ForeignKey('user.user_id'))

user = relationship('User', back_populates='devices')
data = relationship('Data', back_populates='device', cascade='all, delete-orphan')

class Data(Base):
    """Represents data entries from devices in the database."""
    __tablename__ = 'data'
    data_id = Column(Integer, primary_key=True, autoincrement=True)
    device_id = Column(String(100), ForeignKey('device.device_id'))
    time = Column(DateTime)
    temp = Column(Float)
    ph = Column(Float)
    tds = Column(Float)
    do = Column(Float)
    orp = Column(Float)
    salinity = Column(Float)
    water_h = Column(Float)
    water_cl = Column(Float)

    device = relationship('Device', back_populates='data')

# Create all tables in the database
Base.metadata.create_all(engine)

# Create a configured "Session" class
Session = sessionmaker(bind=engine)

# Create a scoped session factory
SessionFactory = sessionmaker(bind=engine)
Session = scoped_session(SessionFactory)

def add_user(data):
    """Add a new user to the database."""
    session = Session()
    try:
        # Check if the username or email already exists
        if user_exists(data['username']) or user_exists(data['email']):
            return False

        # Create a new user object
        new_user = User(
            username=data['username'],
            email=data['email'],
            admin=False
        )

        # Hash and set the password
        new_user.set_password(data['password'])

        session.add(new_user)
        session.commit()
        return True
    finally:
        session.close()

def login_user(data):
    """Authenticate a user login based on username and password."""
    session = Session()
    try:
        user = session.query(User).filter_by(username=data['username']).first()
        return bool(user and user.check_password(data['password']))
    finally:
        session.close()

def user_exists(username_or_email):

```

```

"""Check if a user or email already exists in the database."""
session = Session()
try:
    return session.query(User).filter(
        (User.username == username_or_email) | (User.email == username_or_email)
    ).first() is not None
finally:
    session.close()

def is_admin(user_id):
    """Check if a user is an admin."""
    session = Session()
    try:
        user = session.query(User).filter_by(user_id=user_id).first()
        if user:
            return user.admin
        return False
    finally:
        session.close()

def update_user_info(old_user_id, new_username, new_email, new_password):
    """Update user information in the database."""
    session = Session()
    try:
        # Retrieve the user object from the database
        user = session.query(User).filter_by(user_id=old_user_id).first()

        # If the user is not found, return False
        if not user:
            return False

        # Update user attributes
        user.username = new_username
        user.email = new_email

        if new_password:
            user.set_password(new_password)

        # Flush the changes to ensure the new username is updated in the user table
        session.flush()

        session.commit()

        return True
    except Exception as e:
        print(f"Error updating user info: {e}")
        session.rollback()
        return False
    finally:
        session.close()

def add_device_to_db(user_id, device_id, device_name, latitude, longitude):
    """Add a new device to the database."""
    session = Session()
    try:
        if device_exists(device_id) or device_exists(device_name):
            return False

        new_device = Device(
            device_id=device_id,
            user_id=user_id,
            device_name=device_name,
            latitude=latitude,
            longitude=longitude
        )

        session.add(new_device)

        session.commit()
        return True
    
```

```

        finally:
            session.close()

def update_device_in_db(old_device_id, new_user_id, new_device_name, new_latitude,
new_longitude):
    """Update device information in the database."""
    session = Session()
    try:
        device = session.query(Device).filter_by(device_id=old_device_id).first()
        if not device:
            return False

        device.user_id=new_user_id,
        device.device_name=new_device_name,
        device.latitude=new_latitude
        device.longitude=new_longitude

        # Flush the changes to ensure the new username is updated in the user table
        session.flush()

        session.commit()
        return True
    finally:
        session.close()

def delete_device_by_id(device_id):
    """Delete a device by its ID from the database."""
    session = Session()
    try:
        device = session.query(Device).filter_by(device_id=device_id).first()
        if device:
            session.delete(device)
            session.commit()
            return True
        return False
    except Exception as e:
        print(f"Error deleting device: {e}")
        session.rollback()
        return False
    finally:
        session.close()

def device_exists(device_id_or_device_name):
    """Check if a device already exists in the database."""
    session = Session()
    try:
        return session.query(Device).filter(
            (Device.device_id == device_id_or_device_name) | (Device.device_name ==
device_id_or_device_name)
        ).first() is not None
    finally:
        session.close()

def get_all_devices():
    """Retrieve all devices from the database."""
    session = Session()
    try:
        return session.query(Device).order_by(Device.device_name).all()
    finally:
        session.close()

def get_all_users():
    """Retrieve all users from the database."""
    session = Session()
    try:
        return session.query(User).order_by(User.username).all()
    finally:
        session.close()

def get_user_id(username):

```

```

"""Retrieve user ID by username from the database."""
session = Session()
try:
    user = session.query(User).filter_by(username=username).first()
    return user.user_id
finally:
    session.close()

def get_user_info(user_id):
    """Retrieve user information by user ID from the database."""
    session = Session()
    try:
        return session.query(User).filter_by(user_id=user_id).first()
    finally:
        session.close()

def get_user_devices(user_id):
    """Retrieve devices associated with a specific user from the database."""
    session = Session()
    try:
        return session.query(Device).filter_by(user_id=user_id).order_by(Device.device_name).all()
    finally:
        session.close()

def get_device_data(device_id):
    """Retrieve data entries for a specific device from the database."""
    session = Session()
    try:
        return session.query(Data).filter_by(device_id=device_id).order_by(desc(Data.time)).all()
    finally:
        session.close()

def get_device_last_data(device_id):
    """Retrieve the most recent data entry for a specific device from the database."""
    session = Session()
    try:
        return session.query(Data).filter_by(device_id=device_id).order_by(desc(Data.time)).first()
    finally:
        session.close()

def get_device_info(device_id):
    """Retrieve device information by its ID from the database."""
    session = Session()
    try:
        return session.query(Device).filter_by(device_id=device_id).first()
    finally:
        session.close()

def get_data_by_device_and_time(device_id, time_range):
    """Retrieve data entries for a device within a specified time range from the database."""
    session = Session()
    try:
        # Determine the start time based on the specified time range
        if time_range == 'today':
            start_time = datetime.now().replace(hour=0, minute=0, second=0,
microsecond=0).strftime('%Y-%m-%d %H:%M:%S')
        elif time_range == '24_hours':
            start_time = (datetime.now() - timedelta(hours=24)).strftime('%Y-%m-%d
%H:%M:%S')
        elif time_range == 'this_week':
            start_time = (datetime.now() -
timedelta(days=datetime.now().weekday())).replace(hour=0, minute=0, second=0,
microsecond=0).strftime('%Y-%m-%d %H:%M:%S')
        elif time_range == 'one_week':
            start_time = (datetime.now() - timedelta(weeks=1)).strftime('%Y-%m-%d
%H:%M:%S')
    finally:
        session.close()

```

```

        elif time_range == 'all_time':
            start_time = datetime.min.strftime('%Y-%m-%d %H:%M:%S') # Set to a very
old date
        else:
            start_time = (datetime.now() - timedelta(weeks=1)).strftime('%Y-%m-%d
%H:%M:%S')

        # Query data from the database based on the device ID and time range
        data_objects = session.query(Data).filter(
            Data.device_id == device_id,
            Data.time >= start_time
        ).order_by(Data.time).all()

        # Initialize an empty list to store dictionaries
        data_dicts = []

        # Convert data objects to dictionaries
        for data_obj in data_objects:
            data_dict = {
                'time': data_obj.time.strftime('%Y-%m-%d %H:%M:%S'),
                'temp': data_obj.temp,
                'ph': data_obj.ph,
                'tds': data_obj.tds,
                'do': data_obj.do,
                'orp': data_obj.orp,
                'salinity': data_obj.salinity,
                'water_h': data_obj.water_h,
                'water_cl': data_obj.water_cl
            }
            data_dicts.append(data_dict)

        # Serialize to JSON
        return json.dumps(data_dicts)
    finally:
        session.close()

def store_data(device_id):
    """Fetch and store data from an external API to the database."""
    session = Session()
    try:
        # Retrieve device name and last recorded time from the database
        device_name = get_device_info(device_id=device_id).device_name
        try:
            last_time = get_device_last_data(device_id=device_id).time
        except AttributeError:
            last_time = datetime(1970, 1, 1)

        # Define the URL and headers for the external API
        url = f"https://platform.antares.id:8443/~/antares-cse/antares-
id/{app_lora}/{device_name}?fu=1&drt=2&ty=4"
        headers = {
            "X-M2M-Origin": f"{api_key}",
            "Content-Type": "application/json",
            "Accept": "application/json"
        }

        # Fetch data from the API
        json_array = fetch_data_api_antares(device_id, url, headers, last_time)
        if json_array:
            # Iterate over fetched data and add to the database
            for entry in json_array:
                new_data = Data(
                    device_id=entry['device_id'],
                    time=entry['time'],
                    temp=entry['temp'],
                    ph=entry['ph'],
                    tds=entry['tds'],
                    do=entry['do'],
                    orp=entry['orp'],
                    salinity=entry['salinity'],
                    water_h=entry['water_h'],
                    water_cl=entry['water_cl']
                )
                session.add(new_data)
    finally:
        session.commit()

```

```
        water_h=entry['water_h'],
        water_cl=entry['water_cl']
    )
    session.add(new_data)

    # Commit the changes to the database
    try:
        session.commit()
        print("Data successfully stored in the database.")
    except IntegrityError as e:
        session.rollback()
        print(f"Error inserting data: {e}")
finally:
    session.close()
```

LAMPIRAN F KODE PROGRAM BACKEND FETCH DATA ANTARES

```
"""
fetch_data.py

This module contains functions for interacting with the Antares API,
including fetching and processing data from connected devices. It includes
functions for decompressing data received from the API using the Unishox2
compression algorithm and converting it into a usable format. The main
functionalities include handling API requests, parsing and filtering
responses, and ensuring data integrity for further processing in the
application.
"""

import ctypes
import json
from datetime import datetime
import requests

def decompress(data):
    """
    Decompresses compressed data using the Unishox2 decompression library.

    Args:
        data (str): Hexadecimal string of compressed data.

    Returns:
        dict: Decompressed data as a Python dictionary.
    """
    # Load the shared library (decompress library)
    lib = ctypes.CDLL('./libunishox2.dll') # Change to 'libunishox2.dll' on Windows

    # Define the Unishox2 decompress function signature
    lib.unishox2_decompress_simple.argtypes = [ctypes.c_char_p, ctypes.c_int, ctypes.c_char_p]
    lib.unishox2_decompress_simple.restype = ctypes.c_int

    compressed_data = bytes.fromhex(data)

    # Allocate a buffer for the decompressed data
    decompressed_data = ctypes.create_string_buffer(1024) # Adjust size as needed

    # Decompress the data
    decompressed_length = lib.unishox2_decompress_simple(
        compressed_data,
        len(compressed_data),
        decompressed_data
    )

    # Get the decompressed data as a string
    decompressed_string = decompressed_data.raw[:decompressed_length].decode('utf-8')

    # Convert the JSON string to a Python dictionary
    data_dict = json.loads(decompressed_string)

    return data_dict

def fetch_data_api_antares(device_id, url, headers, last_time):
    """
    Fetches data from the Antares API.

    Args:
        device_id (str): Device ID.
        url (str): URL of the API endpoint.
        headers (dict): Headers for the API request.
        last_time (datetime.datetime): Last processed time.

    Returns:
        list: List of JSON objects containing fetched data.
    """

```

```

"""
# Send GET request with timeout
response = requests.get(url, headers=headers, timeout=10)
# Create an array to store the JSON objects
json_array = []
# Check if request was successful
if response.status_code == 200:
    # Parse JSON response
    data = response.json()
    # Filter entries with the current date (only date part)
    for entry in data["m2m:list"]:
        # Get the creation time
        creation_time_str = entry["m2m:cin"]["ct"]
        creation_time = datetime.strptime(creation_time_str, "%Y%m%dT%H%M%S")
        # Continue to the next entry if the creation time is before the last processed
time
        if creation_time <= last_time:
            continue
        con_data = entry["m2m:cin"]["con"]
        # Check if the data is JSON
        try:
            con_data_json = json.loads(con_data)
            # Check if the data contains a "data" key
            if "data" in con_data_json:
                data_value = con_data_json["data"]
                # Check if data needs to be decompressed
                decompressed_data = decompress(data_value)

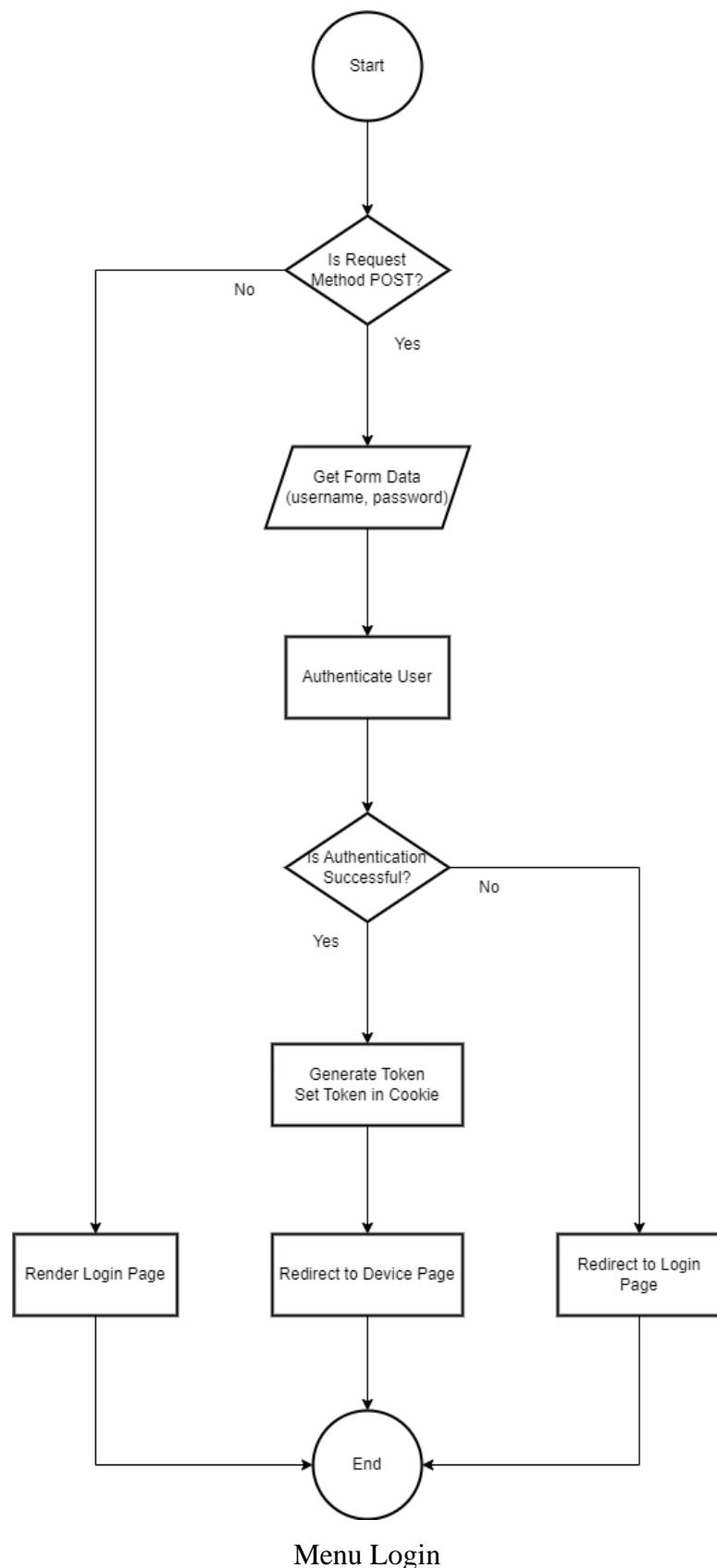
                json_object = {
                    'time': creation_time,
                    'device_id': device_id,
                    'temp': decompressed_data.get('temp', None),
                    'ph': decompressed_data.get('ph', None),
                    'tds': decompressed_data.get('tds', None),
                    'do': decompressed_data.get('do', None),
                    'orp': decompressed_data.get('orp', None),
                    'salinity': decompressed_data.get('salinity', None),
                    'water_h': decompressed_data.get('water_h', None),
                    'water_cl': decompressed_data.get('water_cl', None)
                }

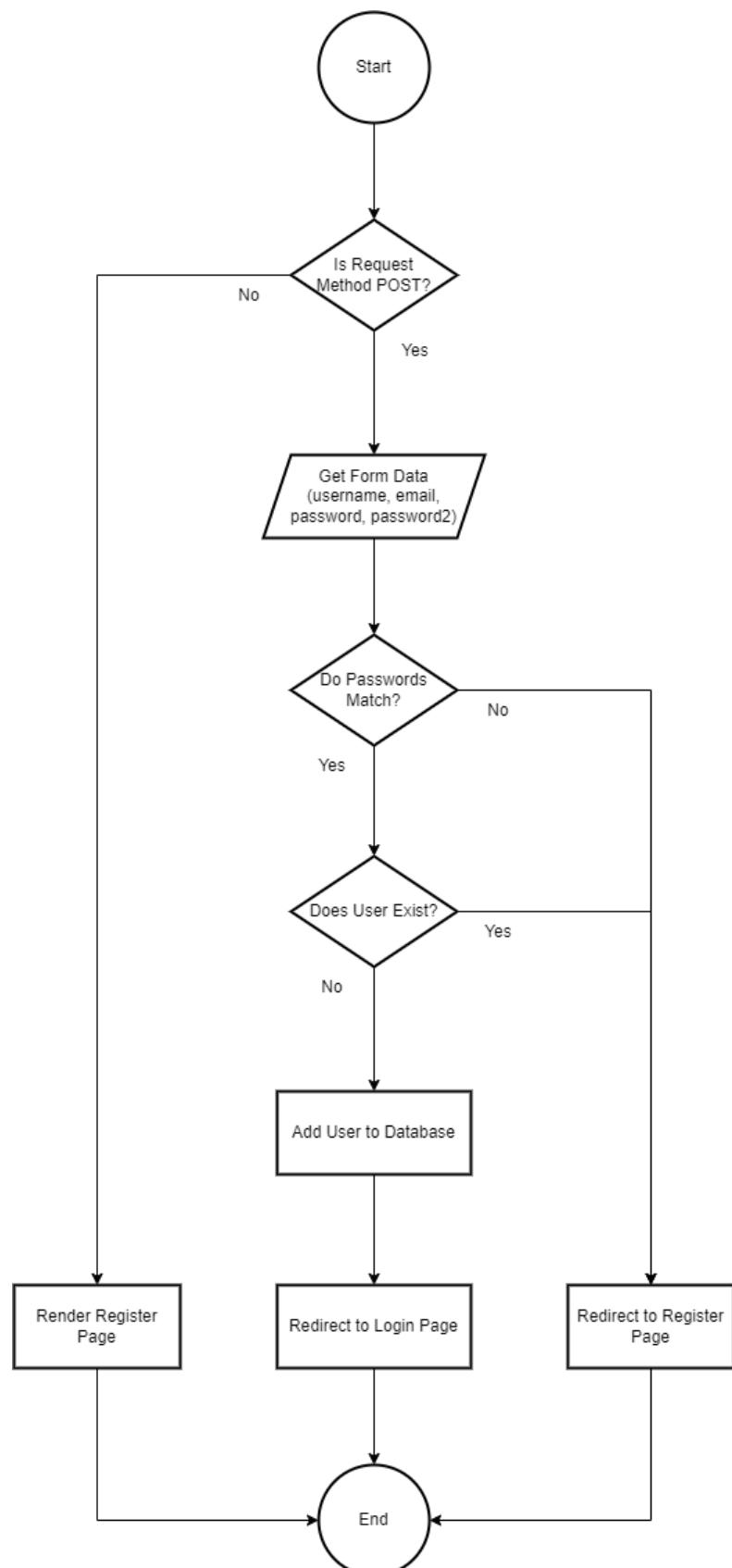
                # Append the JSON object to the array
                json_array.append(json_object)
            else:
                print("Skipping data without 'data' key:", con_data)
        except json.JSONDecodeError:
            print("Skipping non-JSON data:", con_data)
            print(f"Successfully fetched {len(json_array)} data points from device
{device_id}.")
        else:
            print("Failed to fetch data. Status code:", response.status_code)

return json_array

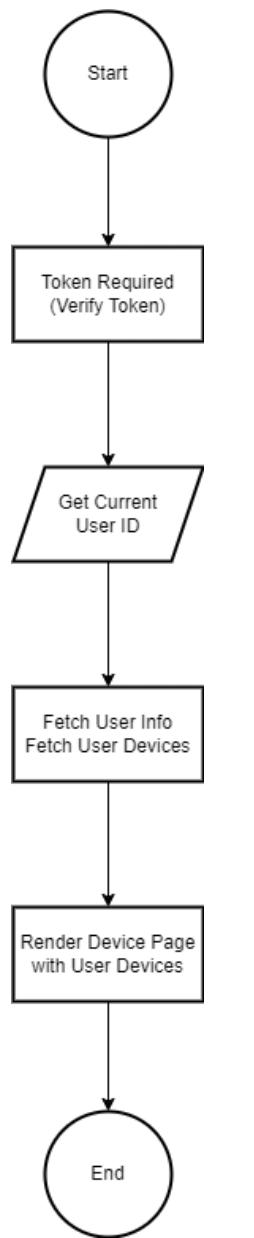
```

LAMPIRAN G FLOWCHART MENU DAN FITUR

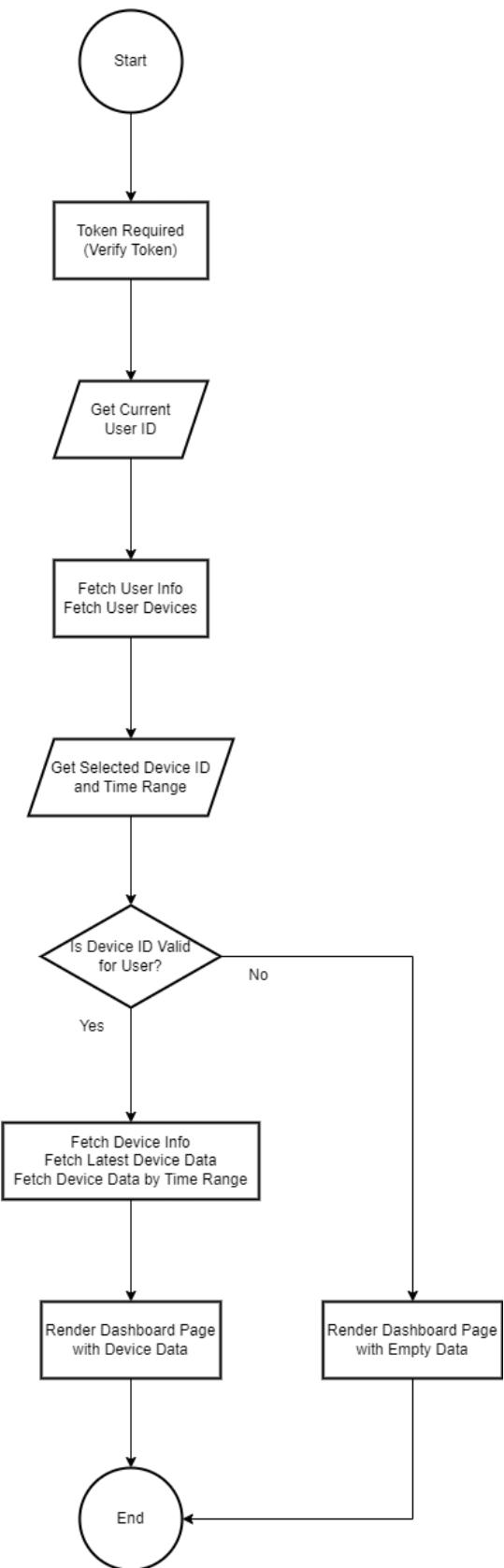




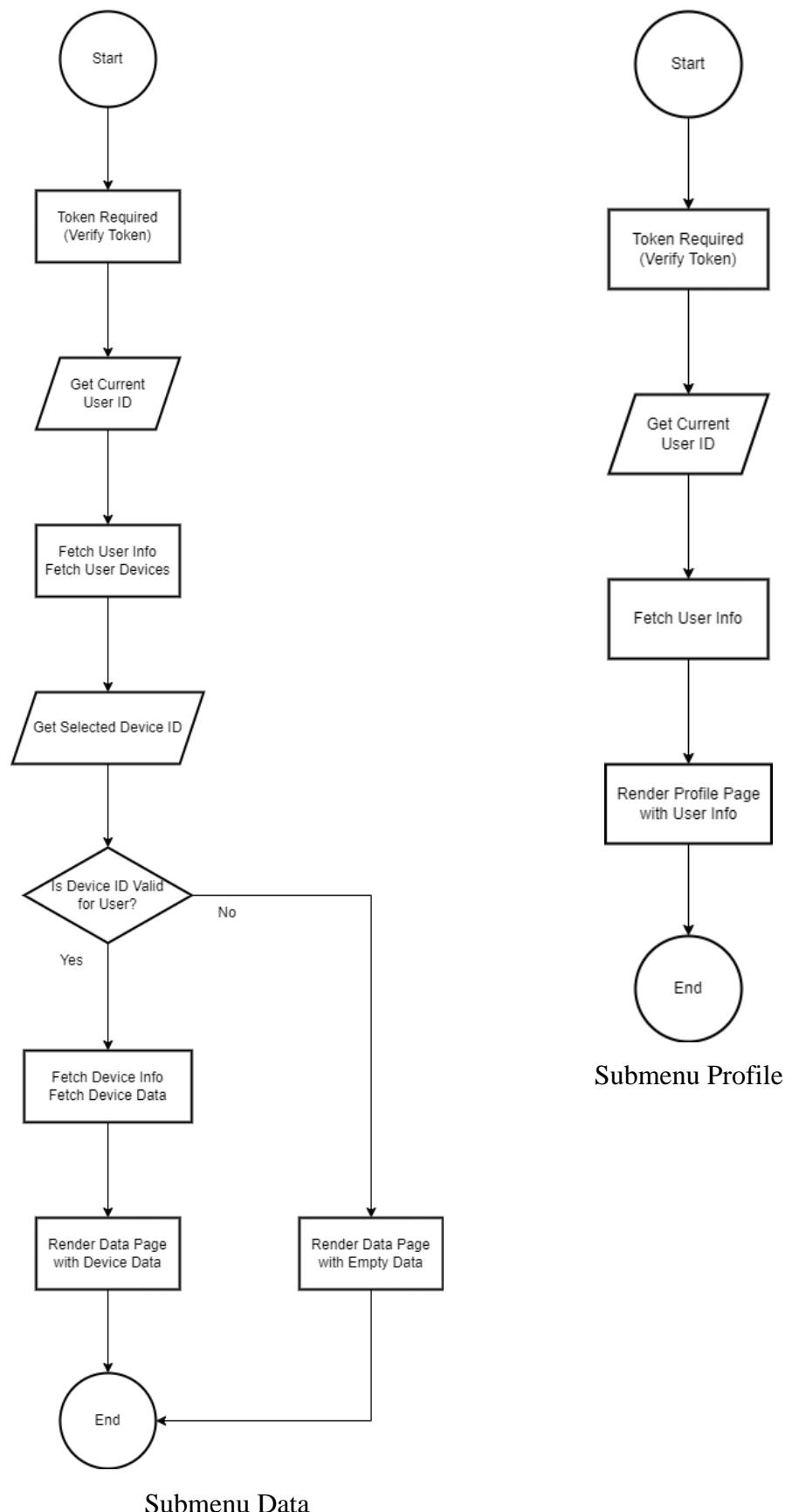
Menu Register

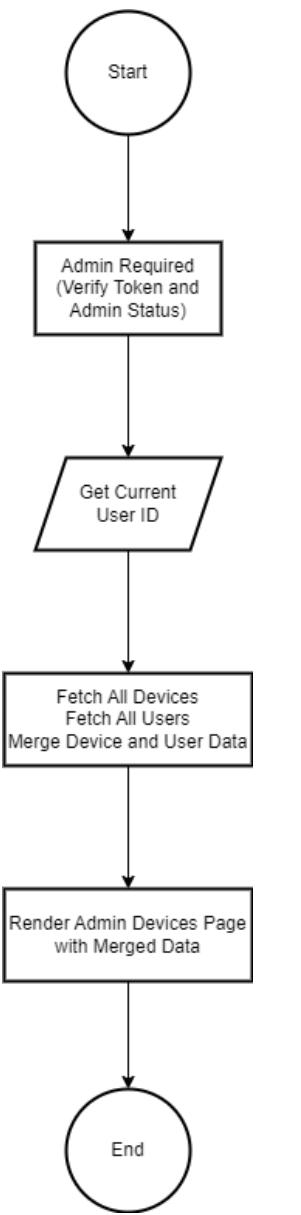


Submenu Device

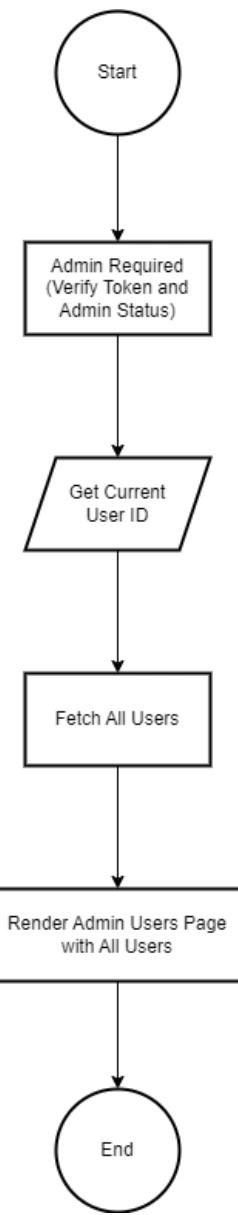


Submenu Dashboard

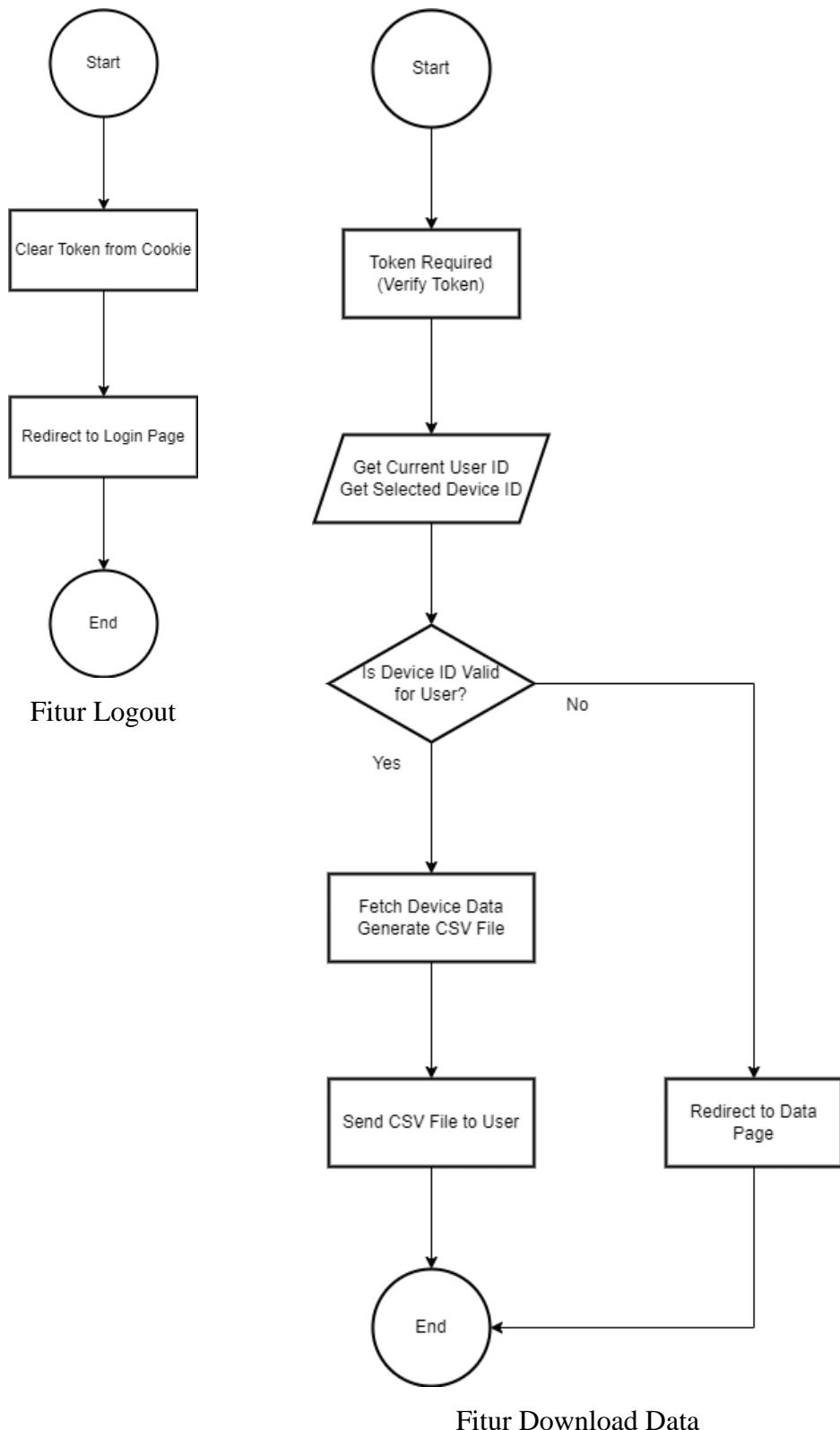


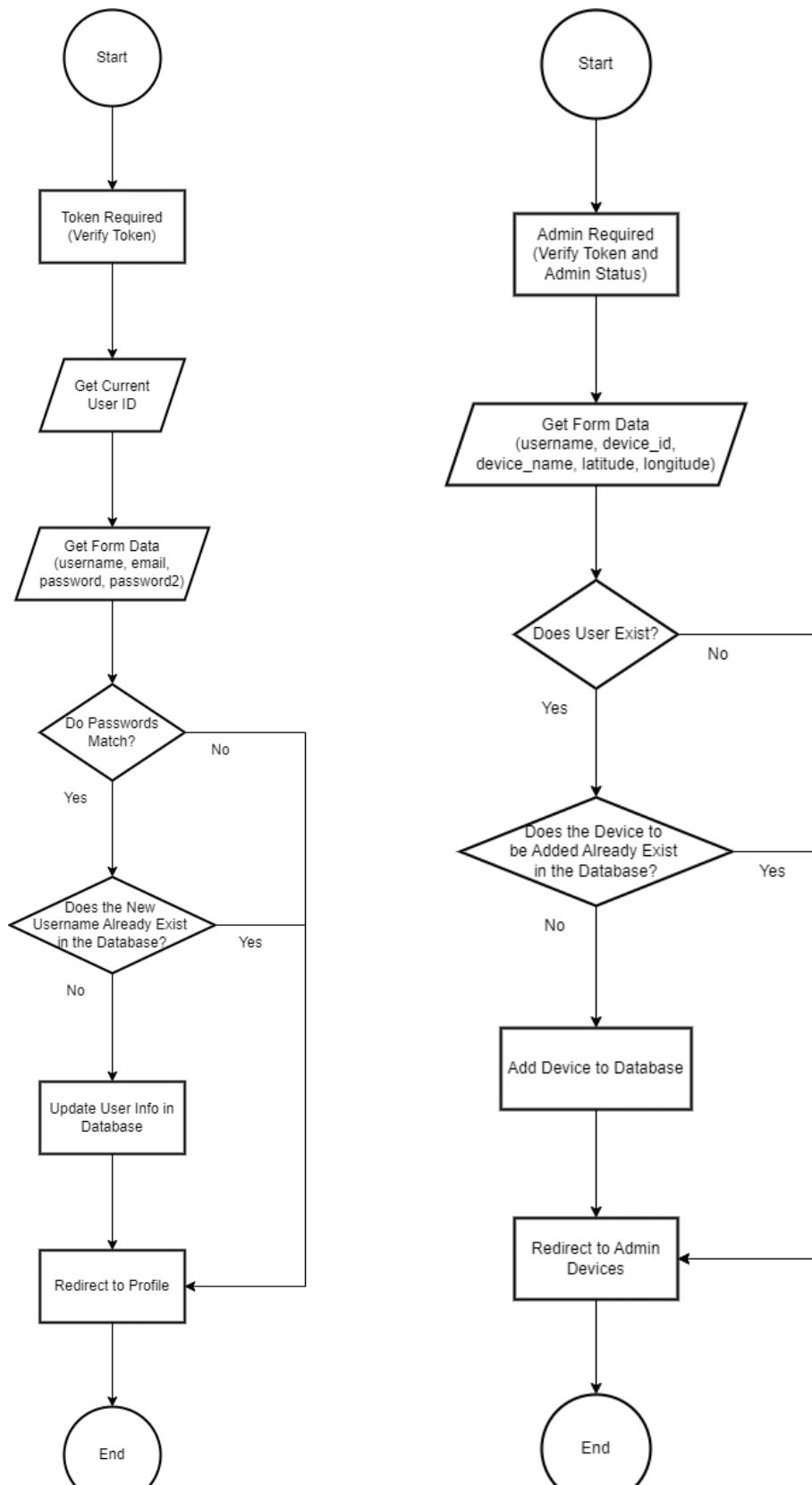


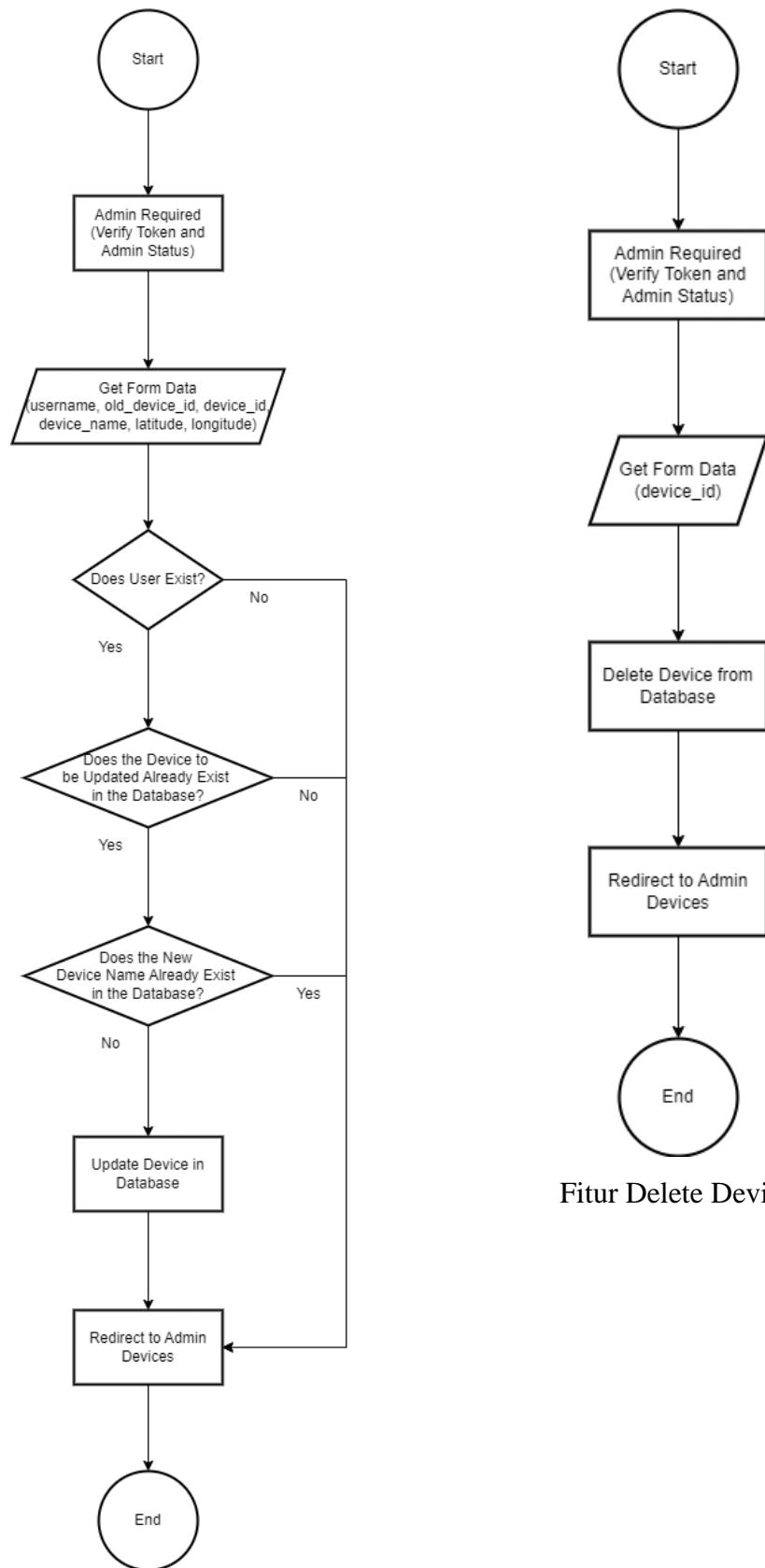
Submenu Admin Devices



Submenu Admin Users







Fitur Edit Device

Fitur Delete Device