

SISTEM MONITORING SMART *GREENHOUSE*
BERBASIS IOT DENGAN MENGGUNAKAN METODE
AGILE KANBAN
(Studi Kasus *Greenhouse Melon* ITERA)
TUGAS AKHIR

Diajukan sebagai syarat menyelesaikan jenjang strata Satu (S-1)
di Program Studi Teknik Informatika, Jurusan Teknologi,
Produksi dan Industri, Institut Teknologi Sumatera

Oleh :

Rafi Arya Nugraha
119140060



PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI, PRODUKSI DAN INDUSTRI
INSTITUT TEKNOLOGI SUMATERA
LAMPUNG SELATAN
2023

LEMBAR PENGESAHAN

Tugas Akhir dengan judul “Sistem Monitoring Smart *Greenhouse* Berbasis IoT Dengan Menggunakan Metode Agile Kanban (Studi Kasus Greenhouse Melon ITERA)” adalah benar dibuat oleh saya sendiri dan belum pernah dibuat dan diserahkan sebelumnya, baik sebagian ataupun seluruhnya, baik oleh saya ataupun orang lain, baik di Institut Teknologi Sumatera maupun di institusi pendidikan lainnya.

Lampung Selatan, 06-08-2023

Penulis,

PHOTO
BERWARNA

Rafi Arya Nugraha

NIM. 119140060

Diperiksa dan disetujui oleh,

Pembimbing

Tanda Tangan

1. Muhammad Habib Algifari, S.Kom., M.T.I.
NIP. 19910525 2022 03 1 002

.....

2. Ilham Firman Ashari, S.Kom., M.T.
NIP. 19930314 201903 1 018

.....

Pengaji

Tanda Tangan

1. Eko Dwi Nugroho, S.Kom., M.Cs.
NRK. 19910209 2020 1 279

.....

2. Aidil Afriansyah, S.Kom., M.Kom.
NIP. 19910416 201903 1 015

.....

Disahkan oleh,
Koordinator Program Studi Teknik Informatika
Jurusan Teknologi, Produksi dan Industri
Institut Teknologi Sumatera

Andika Setiawan, S.Kom., M.Cs.
NIP. 19911127 2022 03 1 007

HALAMAN PERNYATAAN ORISINALITAS

Tugas Akhir dengan judul “Sistem Monitoring Smart *Greenhouse* berbasis IoT dengan Menggunakan Metode Agile Kanban (Studi Kasus *Greenhouse Melon ITERA*)” adalah karya saya sendiri, dan semua sumber baik yang dikutip maupun dirujuk telah saya nyatakan benar.

Nama :

NIM :

Tanda Tangan :

Tanggal :

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai civitas akademik Institut Teknologi Sumatera, saya yang bertanda tangan di bawah ini:

Nama : Rafi Arya Nugraha
NIM : 119140060
Program Studi : Teknik Informatika
Jurusan : Jurusan Teknologi, Produksi dan Industri
Jenis Karya : Tugas Akhir

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Institut Teknologi Sumatera **Hak Bebas Royalti Noneksklusif (Non-exclusive Royalty Free Right)** atas karya ilmiah saya yang berjudul:

SISTEM MONITORING SMART GREENHOUSE BERBASIS IOT DENGAN MENGGUNAKAN METODE AGILE KANBAN (Studi Kasus Greenhouse Melon ITERA)

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini, Institut Teknologi Sumatera berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan skripsi saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di Lampung Selatan
Pada tanggal 06 Agustus 2023

Yang menyatakan,

Rafi Arya Nugraha

KATA PENGANTAR

Puji syukur penulis ucapkan kepada Allah yang telah memberikan kesehatan dan kekuatan kepada penulis sehingga dapat menyelesaikan penulisan tugas skhir ini yang berjudul "**Sistem Monitoring Smart *Greenhouse* berbasis IoT dengan Menggunakan Metode Agile Kanban (Studi Kasus Greenhouse Melon ITERA)**". Penyusunan tugas akhir ini diajukan sebagai syarat untuk mendapatkan gelar Sarjana di Program Studi Teknik Informatika, Jurusan Teknologi Produksi dan Industri pada Institut Teknologi Sumatera. Tak lupa penulis ucapkan terima kasih karena dalam penyusunan tugas akhir ini tidak lepas dari bantuan beberapa pihak. Maka dari itu penulis ingin menyampaikan ucapan terima kasih kepada pihak yang telah membantu selama penyusunan tugas akhir ini.

Ucapan terima kasih penulis sampaikan dalam membantu pelaksanaan hingga pembuatan laporan diantaranya :

1. Kepada ayah, ibu, dan kakak yang telah mendukung, memberikan semangat, memberikan doa, dan bantuan dalam menyelesaikan tugas akhir ini.
2. Bapak Muhammad Habib Algifari, S.Kom., M.T.I sebagai pembimbing 1 dan juga Bapak Ilham Firman Ashari, S.Kom., M.T selaku pembimbing 2 pada tugas akhir ini yang telah memberikan bimbingan dan masukan untuk dapat menyelesaikan tugas akhir.
3. ibu Zunanik Mufidah, S.TP., M.Si. sebagai pembimbing dari pembuatan *greenhouse* serta Mas Kharisma dan Mas Eren dari PT. Kharisma Agri Inovasi yang telah membantu dalam pembuatan IoT.
4. Jajaran dosen ibu Winda Yulita, M.Cs, ibu Nike Dwi G. Drantantiyas, S.Si., M.T, ibu Raizummi Fil'aini, S.T.P., M.Si, dan juga dosen dari kedaireka *Smart Farming System* lainnya yang belum disebutkan.
5. Teman-teman dari kedaireka yang telah membangun sistem Bersama Aqshal, ilham, farras, dhifaf, eko, izqi, ardiyan, alif, kresna, jidan, fauzan, hanifah, haniyah , monika, irin, partina, aca, aysraf, kiel.
6. Tak lupa kepada teman teman yang telah menyemangati dan menemaninya dalam pembuatan tugas akhir ini ajai, randi, afif, ridho, rauf, hariz, Irfan, chika, tribun, donny, awi, yoas.

Penulis menyadari bahwa laporan ini memiliki banyak sekali kekurangan. Penulis terbuka dan menerima setiap kritik dan saran yang membangun dari semua pihak. Akhir kata, penulis

berharap laporan tugas akhir ini bisa bermanfaat bagi para pembaca, khususnya bagi mahasiswa/i Institut Teknologi Sumatera.

Bandar Lampung, 06 Aguustus 2023

Rafi Arya Nugraha

119140060

RINGKASAN

Sistem Monitoring Smart *Greenhouse* Berbasis IoT Dengan Menggunakan Metode Agile Kanban

Rafi Arya Nugraha

Latar belakang penelitian ini adalah terjadinya permasalahan penanaman melon yang dilakukan di *greenhouse* Institut Teknologi sumatera. Melon sendiri merupakan salah satu buah diminati oleh banyak masyarakat dan mempunyai harga yang relatif tinggi, baik untuk pasar domestik. Akan tetapi buah melon merupakan buah yang sulit ditanam dibandingkan anggota *Cucurbitaceae* lainnya. Untuk dapat tumbuh dengan baik, melon membutuhkan tanah, iklim, air dan lokasi penanaman yang ideal. Sehingga diperlukan sebuah sistem *smart greenhouse* yang dapat mengatur suhu, kelembaban, dan jumlah nutrisi yang diberikan sehingga dapat memaksimalkan hasil panen dari pembibitan melon. Sistem yang cocok untuk melakukan monitoring *greenhouse* adalah *Internet of Things* sehingga pengguna dapat melakukan monitoring setiap waktu. Data-data yang disajikan seperti suhu, jumlah, nutrisi, dan kelembaban melalui website. Dalam pembuatan IoT terdapat banyak metode yang dapat dilakukan salah satu metode yang cepat dan fleksibel dapat digunakan untuk mengembangkan IoT adalah *agile* dengan menggunakan *framework kanban*. Selain itu sebuah sistem perlu dilakukan pengujian, Pengujian yang dilakukan dalam penelitian ini menggunakan *System Usability Scale* dan juga *black box*. Tujuan dilakukannya penelitian ini adalah untuk mengetahui cara pembuatan *sistem monitoring smart greenhouse* berbasai IoT menggunakan metode *agile kanban*. Kedua mengetahui hasil evaluasi dari *Website*, dan terakhir untuk mengetahui perbedaan antara penanaman melon secara konvensional dan juga menggunakan teknologi IoT.

Dalam penelitian ini dilakukan studi literatur pada jurnal terdahulu mengenai sistem *smart greenhouse* yang telah dibuat sebelumnya sebagai salah satu acuan dalam memilih teknologi yang akan digunakan. Penelitian menggunakan data primer yang didapatkan dari wawancara dan observasi pada pemilik *greenhouse* dan untuk data sekunder berdasarkan jurnal untuk mengetahui kondisi ideal dari *greenhouse*. Pembuatan alat digunakan dengan menggunakan metode *agile kanban* terbagi mulai dari *planning* yang dilakukan pencarian *user story* dan juga kebutuhan fungsional sistem yang akan dibuat berdasarkan data-data yang ada, lalu tahapan selanjutnya adalah *design* yang berisi mengenai pembuatan diagram *uml*

yakni *use case*, *entity relational diagram*, diagram blok, *activity diagram*, desain ui&ux dan desain alat.

Tahap pengembangan dilakukan dengan mengerjakan *backend*, *frontend*, dan sistem tertanam pada tahap ini dilakukan pembuatan sistem secara keseluruhan pengerajan dilakukan berdasarkan rancangan yang telah dibuat sebelumnya, rancangan dilakukan dengan melist kartu-kartu yang akan dikerjakan kartu-kartu tersebut berdasarkan *user story* yang telah dibuat. Metode *agile kanban* dilakukan dengan mengerjakan berdasarkan kartu, jika sistem telah dibuat dan dilakukan pengujian telah berhasil pada *borad kanban* akan dilakukan *update* jika sedang mengerjakan dan telah selesai dikerjakan. Hasil dari tahapan ini adalah alat IoT dengan hasil pengujian *black box*.

Sistem berhasil dibuat tahapan selanjutnya adalah melakukan *deploy* dari sistem atau melakukan *hosting* dari sistem agar sistem dapat diakses di berbagai *device* sehingga sistem dapat dilakukan pengujian berupa *System Usability Scale*. Pengujian dilakukan pada pemilik *greenhouse*, penjaga *greenhouse*, orang yang ahli pada bidang *greenhouse*, dan juga mahasiswa dari Institut Teknologi Sumatera yang memiliki pemahaman terhadap IoT. Hasil dari pengujian yang dilakukan sistem memiliki nilai 75.875 atau dapat diterima oleh pengguna sehingga sistem layak untuk digunakan atau dapat masuk ke tahap selanjutnya yakni *laucnh*.

Pada tahapan *launch* sistem akan dilakukan perbandingan sistem konvensional dan juga sistem menggunakan *IoT* sehingga didapatkan perbedaan hasil penanaman melon didapatkan kesimpulan yakni. Pada tanaman melon yang menggunakan sistem berbasis IoT memiliki diameter buah yang lebih besar dan lebih berat akan tetapi pada tingkat kemanisan melon konvensional lebih manis hal ini dikarenakan masih kurangnya nutrisi yang diberikan pada tanaman melon berbasis IoT. Berdasarkan hal tersebut diperlukan pengembangan selanjutnya yang dapat melakukan otomatisasi pada pengisian tandon nutrisi secara otomatis.

ABSTRAK

Sistem Monitoring Smart *Greenhouse* Berbasis IoT Dengan Menggunakan Metode Agile

Kanban

Rafi Arya Nugraha

Komoditas melon diminati oleh banyak masyarakat dan mempunyai harga yang relatif tinggi, baik untuk pasar domestik. Akan tetapi buah melon merupakan buah yang sulit ditanam dibandingkan anggota *Cucurbitaceae* lainnya. Untuk dapat tumbuh dengan baik, melon membutuhkan tanah, iklim, air dan lokasi penanaman yang ideal. Sehingga diperlukan sebuah sistem *smart greenhouse* yang dapat mengatur suhu, kelembaban, dan jumlah nutrisi yang diberikan sehingga dapat memaksimalkan hasil panen dari pembibitan melon. Sistem yang cocok untuk melakukan monitoring *greenhouse* adalah *Internet of Things* sehingga pengguna dapat melakukan monitoring setiap waktu. Data-data yang disajikan seperti suhu, jumlah, nutrisi, dan kelembaban melalui website yang dibuat menggunakan metode *agile kanban*. Setelah sistem dibuat dilakukan pengujian berupa *black box* testing dan *system usability scale*. Hasil penelitian yang dilakukan didapatkan bahwa sistem berhasil lulus dari pengujian *black box* dan pada *system usability scale* memiliki nilai 75.875 atau dapat diterima oleh pengguna sehingga pembuatan sistem monitoring *smart greenhouse berbasis IoT* berhasil dilakukan dengan menggunakan metode *agile kanban*. Hasil pada tanaman melon yang menggunakan sistem berbasis IoT memiliki diameter buah yang lebih besar dan lebih berat akan tetapi pada tingkat kemanisan melon yang masih dibawah dari metode konvensional yang disebabkan oleh masih manualnya pengecekan tandon yang kosong.

Kata Kunci : *Smart Greenhouse, monitoring, Internet of Things, agile kanban,*

ABSTRACT

IoT-Based Smart *Greenhouse* Monitoring System Using the Agile Kanban Method

Rafi Arya Nugraha

Melon commodities are in demand by many people and have relatively high prices, both for the domestic market. However, melon is a difficult fruit to grow compared to other Cucurbitaceae members. To grow well, melons require ideal soil, climate, water and planting location. So a smart *greenhouse* system is needed that can regulate temperature, humidity, and the amount of nutrients provided so as to maximize the yield of melon nurseries. A suitable system for monitoring the *greenhouse* is the Internet of Things so that users can monitor at any time. Data presented such as temperature, amount, nutrients, and humidity through a website created using the agile kanban method. After the system is made, testing is carried out in the form of black box testing and system usability scale. The results of the research conducted found that the system successfully passed the black box test and on the system usability scale had a value of 75.875 or acceptable to users so that the creation of an IoT-based smart *greenhouse* monitoring system was successfully carried out using the agile kanban method. Result of the *smart system* for plant a melon is bigger and heavier than using tradisional method but the tradisional method has advantage on sweat, the reason is melon in *smart system* there is disadvantage of need check the stock of nutrision before watering with nutrision.

Keywords : *Smart Greenhouse, monitoring, Internet of Things, agile kanban,*

DAFTAR ISI

| | |
|--|------|
| LEMBAR PENGESAHAN | II |
| HALAMAN PERNYATAAN ORISINALITAS | III |
| HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS | IV |
| KATA PENGANTAR | V |
| RINGKASAN | VII |
| ABSTRAK | IX |
| ABSTRACT | X |
| DAFTAR ISI | XI |
| DAFTAR GAMBAR | XIV |
| DAFTAR KODE | XXI |
| DAFTAR LAMPIRAN | XXII |
| BAB I PENDAHULUAN | 1 |
| 1.1 Latar Belakang | 1 |
| 1.2 Rumusan Masalah | 5 |
| 1.3 Tujuan Penelitian | 5 |
| 1.4 Batasan Masalah | 5 |
| 1.5 Manfaat Penelitian | 6 |
| 1.6 Sistematika Penulisan | 6 |
| 1.6.1 Bab I Pendahuluan | 6 |
| 1.6.2 Bab II Tinjauan Pustaka | 6 |
| 1.6.3 Bab III Metodologi Penelitian | 6 |
| 1.6.4 Bab IV Hasil dan Pembahasan | 6 |
| 1.6.5 Bab V Kesimpulan dan Saran | 7 |
| BAB II LANDASAN TEORI | 8 |
| 3.1 Tinjauan Pustaka | 8 |

| | |
|--|----|
| 3.2 Dasar Teori | 13 |
| 2.2.1 Smart <i>Greenhouse</i> | 13 |
| 2.2.2 Sistem Monitoring | 14 |
| 2.2.3 Internet of Things | 15 |
| 2.2.4 Otomatisasi dan Penjadwalan | 16 |
| 2.2.5 Mikrokontroller | 18 |
| 2.2.6 Sensor | 19 |
| 2.2.7 Aktuator | 21 |
| 2.2.8 Jaringan Komunikasi | 23 |
| 2.2.9 Website | 24 |
| 2.2.10 Front End | 25 |
| 2.2.11 Backend | 26 |
| 2.2.12 Database | 27 |
| 2.2.13 Framework | 27 |
| 2.2.14 Agile Kanban | 27 |
| 2.2.15 Black Box Testing | 31 |
| 2.2.16 Pengujian | 32 |
| BAB III METODOLOGI PENELITIAN | 34 |
| 3.1 Alur Penelitian | 34 |
| 3.2 Penjabaran Langkah Penelitian | 35 |
| 3.2.1 Identifikasi Masalah | 35 |
| 3.2.2 Pengumpulan Data | 35 |
| 3.2.3 Implementasi Sistem dengan Metode Agile Kanban | 38 |
| 3.2.4 Penarikan Kesimpulan | 39 |
| 3.3 Alat dan Bahan | 39 |
| 3.4 Metode Pengembangan | 40 |
| 3.4.1 Planning | 41 |

| | |
|--|------------|
| 3.4.2 Design | 48 |
| 3.4.3 Develop | 70 |
| 3.4.4 Testing (Pengujian) | 70 |
| 3.4.4 Deploy | 71 |
| 3.4.4 Review Klien | 71 |
| 3.4.4 Launch | 71 |
| 3.5 Rancangan Pengujian | 71 |
| BAB IV HASIL DAN PEMBAHASAN | 75 |
| 4.1 Impelementasi Sistem | 75 |
| 4.1.1 Pembuatan Backend | 77 |
| 4.1.2 Implementasi Front End | 93 |
| 4.1.3 Impementasi Alat | 127 |
| 4.2 Deploy | 145 |
| 4.3 Pengujian SUS | 145 |
| 4.4 Launch | 146 |
| 4.4.1 Kondisi Suhu | 146 |
| 4.4.2 Kondisi Kelembaban | 147 |
| 4.4.3 Kondisi Pemberian Nutrisi | 148 |
| BAB V KESIMPULAN | 152 |
| 5.1 Kesimpulan | 152 |
| 5.2 Saran | 152 |
| DAFTAR PUSTAKA | 153 |
| LAMPIRAN | 158 |

DAFTAR GAMBAR

| | |
|---|----|
| Gambar 2.1 Ilustrasi <i>Greenhouse</i> ITERA | 13 |
| Gambar 2.2 Tampak depan dari <i>Greenhouse</i> ITERA | 14 |
| Gambar 2.3 Penjadwalan metode klasik | 17 |
| Gambar 2.4 Penjadwalan metode Delay Time | 17 |
| Gambar 2.5 Penjadwalan metode Countdown | 18 |
| Gambar 2.6 Esp-8266 | 19 |
| Gambar 2.7 Sensor DHT 11 | 20 |
| Gambar 2.8 Sensor Debit | 20 |
| Gambar 2.9 Selenoid Valve | 21 |
| Gambar 2.10 Relay | 22 |
| Gambar 2.11 Macam-macam konektivitas IoT | 23 |
| Gambar 2.12 Tahapan SDLC metode Agile | 28 |
| Gambar 2.13 Kanban Simple board | 29 |
| Gambar 2.14 Detailed Kanban Board | 29 |
| Gambar 2.15 Alur Agile Kanban | 31 |
| Gambar 2.16 Skala Interpretasi SUS | 33 |
| Gambar 3.1 Alur Penelitian | 34 |
| Gambar 3.2 Ilustrasi <i>Greenhouse</i> ITERA | 36 |
| Gambar 3.3 Alur Agile Kanban | 41 |
| Gambar 3.4 Rancangan Pembuatan IoT pada <i>Greenhouse</i> ITERA | 42 |
| Gambar 3.5 Rangan Greenhouse pada bagian amping kanan | 43 |
| Gambar 3.6 Rangan Greenhouse pada bagian amping kanan | 43 |
| Gambar 3.7 Diagram Blok | 48 |
| Gambar 3.8 Diagram Use Case | 49 |
| Gambar 3.9 Activity Diagram Login | 52 |
| Gambar 3.10 Activity Diagram Monitoring Sensor | 53 |
| Gambar 3.11 Activity Diagram Kendali <i>Greenhouse</i> | 54 |
| Gambar 3.12 Activity Diagram Kendali Otomatis | 55 |
| Gambar 3.13 Desain UI/UX Halaman Login | 56 |
| Gambar 3.14 Desain UI/UX Halaman Dashboard Sensor | 57 |
| Gambar 3.15 Desain UI/UX Grafik Sensor | 57 |

| | |
|---|----|
| Gambar 3.16 Desain UI/UX Halaman Dashboard Actuator | 58 |
| Gambar 3.17 Desain UI/UX Halaman Otomatisasi Manual | 59 |
| Gambar 3.18 Desain UI/UX Halaman Otomatis | 59 |
| Gambar 3.19 Desain UI/UX Halaman Otomatis Tambah Skenario By Sensor | 60 |
| Gambar 3.20 Desain UI/UX Halaman Otomatis Tambah Skenario Penjadwalan | 60 |
| Gambar 3.21 Desain UI/UX Halaman Penambahan Skenario Penjadwalan | 61 |
| Gambar 3.22 Desain UI/UX Halaman Otomatis Berhasil menambahkan lebih dari satu Skenario | 62 |
| Gambar 3.23 Desain UI/UX Halaman Catatan Aktuator | 63 |
| Gambar 3.24 Desain UI/UX Halaman Menu <i>Greenhouse</i> | 63 |
| Gambar 3.25 Desain UI/UX Halaman Tambah <i>Greenhouse</i> | 64 |
| Gambar 3.26 Desain UI/UX Halaman Edit <i>Greenhouse</i> | 64 |
| Gambar 3.27 Desain UI/UX Halaman Menu Monitoring | 65 |
| Gambar 3.28 Desain UI/UX Halaman Edit <i>Greenhouse</i> | 66 |
| Gambar 3.29 Desain UI/UX Halaman Edit Monitoring | 66 |
| Gambar 3.30 Desain UI/UX Halaman Menu Actuator | 67 |
| Gambar 3.31 Desain UI/UX Halaman Tambah Actuator | 67 |
| Gambar 3.32 Desain UI/UX Halaman Edit Aktuator | 68 |
| Gambar 3.33 Desain UI/UX Halaman Notifikasi | 68 |
| Gambar 3.34 Rancangan <i>Cashing box</i> hitam tampak atas | 69 |
| Gambar 3.35 Rancangan <i>Cashing box</i> hitam tampak samping | 69 |
| Gambar 3.36 <i>Cashing box</i> hitam tampak bawah | 69 |
| Gambar 3.37 Rancangan <i>Cashing box</i> tampak bawah pada pengendalian nutrisi | 70 |
| Gambar 3.38 Rancangan <i>Cashing box</i> tampak atas pada pengendalian nutrisi | 70 |
| Gambar 4.1 Spesifikasi umum pembuatan sistem pada Agile Kanban | 75 |
| Gambar 4.2 User Story Agile Kanban | 76 |
| Gambar 4.3 Board Agile Kanban | 77 |
| Gambar 4.4. Basis Data Sistem | 78 |
| Gambar 4.5. Tampilan Front End halaman Login | 93 |
| Gambar 4.6 Tampilan Front end Dashboard | 94 |
| Gambar 4.7 Tampilan Front End Jika Login gagal | 95 |
| Gambar 4.8 Tampilan Front End halaman Dashboard | 96 |
| Gambar 4.9 Tampilan Front end halaman monitoring | 97 |
| Gambar 4.10 Tampilan Front End halaman menambah <i>greenhouse</i> | 97 |

| | |
|--|-----|
| Gambar 4.11 Tampilan jika tidak mengisi data pada tambah <i>greenhouse</i> | 98 |
| Gambar 4.12 Mengisi seluruh field menambah <i>greenhouse</i> | 98 |
| Gambar 4.13. Tampilan jika berhasil menambahkan <i>greenhouse</i> | 99 |
| Gambar 4.14 Tampilan Front End mengedit <i>greenhouse</i> | 100 |
| Gambar 4.15. Tampilan Mengubah field lokasi <i>greenhouse</i> | 100 |
| Gambar 4.16. Hasil mengedit lokasi <i>greenhouse</i> | 101 |
| Gambar 4.17 Peringatan saat menghapus <i>greenhouse</i> | 101 |
| Gambar 4.18 Halaman <i>greenhouse</i> setelah <i>greenhouse</i> di hapus | 102 |
| Gambar 4.19 Tampilan Front End halaman monitoring | 103 |
| Gambar 4.20 Tampilan menambahkan sensor | 104 |
| Gambar 4.21 Tampilan mensubmit data monitoring tanpa data | 104 |
| Gambar 4.22 Tampilan jika berhasil menambahkan sensor | 105 |
| Gambar 4.23 Tampilan setelah menambahkan sensor | 105 |
| Gambar 4.24 Tampilan front end halaman edit sensor | 106 |
| Gambar 4.25 Mengedit range max dan gambar sensor | 106 |
| Gambar 4.26 Perubahan setelah sensor diedit | 107 |
| Gambar 4.27 Peringatan menghapus sensor | 107 |
| Gambar 4.28 Hasil setelah berhasil menghapus sensor | 108 |
| Gambar 4.29 Tampilan front end halaman controlling | 109 |
| Gambar 4.30 Tampilan jika tidak mengisi saat ingin menambah aktuator | 110 |
| Gambar 4.31 Menginput seluruh data untuk menambah aktuator | 110 |
| Gambar 4.32 Tampilan setelah menambahkan aktuator | 111 |
| Gambar 4.33 Tampilan front end mengedit aktuator | 111 |
| Gambar 4.34 Tampilan front end mengedit aktuator | 112 |
| Gambar 4.35 Tampilan hasil mengedit aktuator | 113 |
| Gambar 4.36 Peringatan menghapus aktuator | 113 |
| Gambar 4.37 Tampiilan setelah menghapus aktuator | 114 |
| Gambar 4.38 Tampilan front end halaman notifikasi | 115 |
| Gambar 4.39 Peringatan menghapus riwayat notifikasi | 116 |
| Gambar 4.40 Tampilan hasil menghapus notifikasi | 116 |
| Gambar 4.41 Tampilan front end halaman pembacaan Sensor | 118 |
| Gambar 4.42 Tampilan front end halaman grafik perjam | 119 |
| Gambar 4.43 Tampilan front end halaman grafik perminggu | 119 |
| Gambar 4.44 Tampilan front end halaman grafik perbulan | 120 |

| | |
|--|-----|
| Gambar 4.45 Tampilan front end halaman grafik pertahun | 121 |
| Gambar 4.46 Mengunduh File csv pada data harian | 121 |
| Gambar 4.47 Hasil Mengunduh file CSV | 122 |
| Gambar 4.48 Tampilan front end halaman pengendalian actuator | 123 |
| Gambar 4.49 Tampilan front end data historis status aktuator | 124 |
| Gambar 4.50 Tampilan front end pengaturan otomatis aktuator | 125 |
| Gambar 4.51 Tampilan Front end halaman otomatis actuator | 125 |
| Gambar 4.52 Tampilan Front end jika menyalakan otomatis | 126 |
| Gambar 4.53 Hasil pembuatan alat pada <i>Greenhouse ITERA</i> | 127 |
| Gambar 4.54 Implementasi Sensor DHT 11 | 128 |
| Gambar 4.55 Sensor Debit | 130 |
| Gambar 4.56. Hasil Pembuatan <i>Solenoid Valve</i> | 132 |
| Gambar 4.57 Rincian otomatis solenoid valve berdasarkan waktu | 134 |
| Gambar 4.58 Rincian otomatis solenoid valve berdasarkan sensor debit | 134 |
| Gambar 4.59 Implementasi alat Cooling Pad | 136 |
| Gambar 4.60 Rincian otomatis yang digunakan pada cooling pad | 137 |
| Gambar 4.61 Implementasi alat popma nutrisi | 139 |
| Gambar 4.62 Rincian input otomatis penjadwalan pada pompa | 141 |
| Gambar 4.63 Rincian input otomatis berdasarkan sensor debit pada pompa | 141 |
| Gambar 4.64 Bentuk Aktuator <i>Exhaust Fan</i> | 142 |
| Gambar 4.65 Ukuran gelas yang digunakan untuk metode konvensional | 149 |
| Gambar 4.66 <i>refractometer</i> alat ukur tingkat kemanisan melon | 149 |

DAFTAR TABEL

| | |
|---|----|
| Tabel 2.1 Penelitian Terdahulu | 9 |
| Tabel 2.2 Datasheet ESP8266 | 19 |
| Tabel 2.3 Datasheet DHT 11 | 20 |
| Tabel 2.4 Datasheet Sensor Debit | 21 |
| Tabel 2.5 Datasheet Selenoid Valve | 22 |
| Tabel 2.6 Datasheet Selenoid Valve | 22 |
| Tabel 2.7 Pengujian SUS | 32 |
| Tabel 3.1 Perubahan Suhu pada <i>Greenhouse</i> | 36 |
| Tabel 3.2 Perubahan Kelembaban pada <i>Greenhouse</i> | 37 |
| Tabel 3.3 Perubahan Debit Air pada <i>Greenhouse</i> | 37 |
| Tabel 3.4 Hubungan antara Sensor dan Kendali | 38 |
| Tabel 3.5 Spesifikasi Laptop | 39 |
| Tabel 3.6 Perangkat Lunak yang digunakan | 39 |
| Tabel 3.7 Bahan Penelitian | 40 |
| Tabel 3.8 User Story | 44 |
| Tabel 3.9 Kebutuhan Fungsional | 46 |
| Tabel 3.10 Non Fungsional | 47 |
| Tabel 3.11 Rancangan Pengujian Black box | 71 |
| Tabel 3.12 Pengujian Preangkat Keras | 73 |
| Tabel 4.1 Rincian hasil pembuatan basis data | 78 |
| Tabel 4.2 Rincian API endpoint user | 83 |
| Tabel 4.3 Rincian API endpoint <i>greenhouse</i> | 83 |
| Tabel 4.4 Rincian API Endpoint Sensor | 84 |
| Tabel 4.5 Rincian API endpoint actuator | 86 |
| Tabel 4.6 Rincian API endpoint notifikasi | 87 |
| Tabel 4.7 Rincian API endpoint icon | 88 |
| Tabel 4.8 Rincian API endpoint automation | 88 |
| Tabel 4.9 Rincian API endpoint scheduling | 90 |
| Tabel 4.10 Rincian API endpoint <i>actuator-log</i> | 91 |
| Tabel 4.11 Rincian API endpoint <i>sensor-log</i> | 92 |
| Tabel 4.12 Rincian endpoint dashboard | 92 |
| Tabel 4.13 Rincian API endpoint grafik | 93 |
| Tabel 4.14. Pengujian Black box pada halaman login | 95 |

| | |
|--|-----|
| Tabel 4.15 Pengujian black box pada halaman Dashboard | 96 |
| Tabel 4.16 Rincian pengujian black box pada halaman <i>greenhouse</i> | 102 |
| Tabel 4.17 Pengujian black box pada halaman <i>monitoring</i> | 108 |
| Tabel 4.18 Rincian pengujian black box halaman <i>controlling</i> | 114 |
| Tabel 4.19 Rincian pengujian black box halaman notifikasi | 117 |
| Tabel 4.20 Rincian pengujian black box halman pembacaan sensor | 122 |
| Tabel 4.21 Rincian pengujian black box pada halaman pengendalian aktuator | 124 |
| Tabel 4.22 Rincian pengujian black box halaman otomatis | 126 |
| Tabel 4.23 Pengujian nilai sensor DHT11 | 128 |
| Tabel 4.24 Pengjian status kelemban dan suhu | 129 |
| Tabel 4.25 Pengujian blackbox website dan sensor | 129 |
| Tabel 4.26 Pengujian sensor debit | 130 |
| Tabel 4.27 Pengujian status alat dan website sensor debit | 131 |
| Tabel 4.28 Rincian pengujian black box website dan sensor debit | 131 |
| Tabel 4.29 Pengujian pengendalian <i>solenoid valve</i> melalui <i>website</i> | 133 |
| Tabel 4.30 Pengujian kondisi <i>solenoid valve</i> aktual dan <i>website</i> | 133 |
| Tabel 4.31 Pengujian otomatis <i>solenoid valve</i> | 134 |
| Tabel 4.32 Rincian pengujian aktuator <i>solenoid valve</i> dan website | 135 |
| Tabel 4.33 Rincian pengujian perangkat keras | 135 |
| Tabel 4.34 Pengujian website dan pengendalian <i>solenoid valve</i> | 136 |
| Tabel 4.35 Pengujian cooling pad status actual cooling pad dan status website | 137 |
| Tabel 4.36 Rincian pengujian otomatis pada <i>Cooling pad</i> | 138 |
| Tabel 4.37 Rincian pengujian black box dan alat <i>cooling pad</i> | 138 |
| Tabel 4.38 Rincian pengujian alat Cooling Pad | 139 |
| Tabel 4.39. Pengujian pengendalian pompa dari website | 139 |
| Tabel 4.40 Pengujian kondisi actual pompa dan website | 140 |
| Tabel 4.41 Pengujian otomatis website pada pompa | 141 |
| Tabel 4.42 Pengujian respon pengendalian <i>exhaust fan</i> dari website | 142 |
| Tabel 4.43 Pengujian respon website pada status actual exhaust fan | 143 |
| Tabel 4.44 Pengujian otomatis exhaust fan | 144 |
| Tabel 4.45 Rincian pengujian black box website pada exhaust fan | 144 |
| Tabel 4.46 Rincian pengujian alat exhaust fan | 144 |
| Tabel 4.47 Perhitungan SUS | 145 |
| Tabel 4.48 Kondisi suhu <i>greenhouse</i> setelah dilakukan pemasangan IoT | 146 |

| | |
|---|-----|
| Tabel 4.49 Kondisi kelembaban <i>greenhouse</i> setelah dilakukan pemasangan IoT | 147 |
| Tabel 4.50 Kondisi pemberian nutrisi pada <i>greenhouse</i> setelah dilakukan pemasangan IoT | 148 |
| Tabel 4.51 Gambaran perbedaan penanaman melon menggunakan metode konvensional dan <i>smart system</i> | 150 |
| Tabel 4.52 Perbedaan sistem konvensional dan <i>Smart Greenhouse</i> | 150 |

DAFTAR KODE

| | |
|---|----|
| Kode 4.1 Hasil pebuatan schema sensor | 81 |
| Kode 4.2 Hasil Pembuatan Schema Actuator..... | 81 |
| Kode 4.3 Kode untuk Menjalankan backend | 82 |

DAFTAR LAMPIRAN

| | |
|---|-----|
| Lampiran 1 Kode Program | 158 |
| Lampiran 2 <i>Board Agile Kanban</i> | 159 |
| Lampiran 3 Lampiran Pembuatan Backend | 161 |
| Lampiran 4 Pemasangan alat pada <i>smart greenhouse</i> | 163 |
| Lampiran 5 Pengujian alat | 165 |
| Lampiran 6 Pengujian SUS | 165 |
| Lampiran 7 melon metode konvensional | 166 |
| Lampiran 8 melon metode <i>Smart Farming</i> | 167 |

BAB I

PENDAHULUAN

1.1 Latar Belakang

Teknologi dan pendidikan didorong oleh riset yang dikembangkan oleh sebuah perguruan tinggi, salah satu perguruan tinggi di Indonesia mendorong adanya pertanian modern adalah Institut Teknologi Sumatera (ITERA) yang terletak di Lampung Selatan, ITERA mengembangkan Kebun Raya yang lokasinya berada di lingkungan kampus. Kebun raya ITERA yang kini sedang melakukan pembangunan tahap I tersebut diproyeksikan menjadi pusat konservasi tumbuhan khas Pulau Sumatera dan Indonesia serta menjadi pusat penelitian berbagai jenis tumbuhan [1]. Salah satu penelitian dan teknologi yang sedang dikembangkan di kebun raya iter adalah pembuatan *greenhouse* pada tanaman melon, Komoditas melon diminati oleh banyak masyarakat dan mempunyai harga yang relatif tinggi, baik untuk pasar domestik maupun ekspor. Harga buah melon yang relatif tinggi dibanding komoditas sejenis merupakan peluang besar untuk peningkatan pendapatan dan kesejahteraan petani atau pengusaha usahatani melon [2], Akan tetapi buah melon merupakan buah yang sulit ditanam dibandingkan anggota *Cucurbitaceae* lainnya. Untuk dapat tumbuh dengan baik, melon membutuhkan tanah, iklim, air dan lokasi penanaman yang cocok. Tanaman melon yang sehat dan berproduksi optimal berasal dari bibit tanaman yang sehat, kuat dan terawat baik pada awalnya. Oleh karena itu pembibitan merupakan kunci awal bagi keberhasilan agribisnis melon [3]. Melon yang baik dan sehat Menurut PT. East West Seed Indonesia Melon Alisha memiliki ciri-ciri rasa manis dengan (12-16 Brix). umur minimal panen 70 Hari dengan bobot buah 1.5-2.5 kg/buah [4].

Agar mendapatkan buah melon yang tumbuh dengan maksimal terdapat faktor-faktor sehingga buah yang dihasilkan dapat memiliki hasil maksimal, menurut penelitian yang dilakukan oleh I kadek agus indrawan et al bahwa Hasil penelitian menunjukkan bahwa jenis media dan varietas berpengaruh sangat nyata terhadap hasil dari tanaman melon serta dibutuhkan media tanam yang tepat yakni dapat menggunakan *cocopeat* dan aram sekam[5]. Selain varietas dari melon menurut penelitian yang dilakukan oleh Triadiat et al melon ditentukan oleh pemenuhan unsur hara melalui pemupukan, baik dalam penentuan dosis maupun waktu pemberiannya [6].Pemberian pupuk ini sangatlah penting dikarenakan menurut Swaminathan et.al., menyatakan bahwa pemberian pupuk harus dilakukan secara tepat dan sesuai konsentrasi yang dianjurkan, karena pemberian pupuk yang berlebihan akan

menyebabkan keracunan pada tanaman, pertumbuhan tanaman akan terhambat. Dan menimbulkan kerugian, baik kerugian pada pupuk, pada tanaman, maupun pada tanah dan lingkungan di sekitar pemupukan. Selain pembuatan kondisi lingkungan juga menjadi faktor keberhasilan penanaman melon menurut penelitian yang dilakukan oleh Sudaryono bahwa Tanaman melon memerlukan penyiraman matahari penuh selama pertumbuhannya. Tanaman melon memerlukan suhu yang sejuk dan kering untuk pertumbuhannya. Suhu pertumbuhan untuk tanaman melon antara 25-30 °C. Kelembaban udara secara tidak langsung mempengaruhi pertumbuhan tanaman melon. Dalam kelembaban yang tinggi tanaman melon mudah diserang penyakit. sehingga kondisi lingkungan seperti suhu dan kelembaban menjadi faktor yang dapat menyebabkan melon tumbuh dengan tidak maksimal [7].

Sehingga berdasarkan masalah tersebut dapat disimpulkan terdapat faktor yang dapat meningkatkan pertumbuhan melon sehingga menghasilkan buah melon yang diinginkan yakni suhu, kelembaban, intensitas penyiraman dan jumlah nutrisi serta media tanam dari melon sehingga untuk memudahkan diperlukan sebuah solusi yang dapat melakukan monitoring dan kendali pada suhu, jumlah nutrisi, kelembaban, intensitas penyiraman. Dengan membuat sebuah teknologi yang dapat membantu monitoring dan kendali dari *greenhouse* dapat membantu penjaga *greenhouse* dalam merawat kondisi dari *greenhouse* agar melon mendapatkan kondisi yang ideal untuk meminimalisir tanaman yang mati atau pertumbuhannya terhambat, Teknologi yang dapat mendukung berdasarkan masalah diatas adalah konsep Internet of Things. Konsep internet of Things adalah teknologi yang memungkinkan benda-benda di sekitar kita terhubung dengan Internet. Konsep IoT sudah banyak diterapkan seperti alarm rumah, memberikan kondisi cuaca secara real time, membuat Smart Home (Rumah Pintar), SmartParking [8]. Dengan konsep IoT memungkinkan akses dan interaksi yang dapat dilakukan dimana saja dan kapanpun sehingga dalam melakukan monitoring dan kendali pada *greenhouse* dapat dilakukan kapan saja dan dimana saja [9]. Untuk membuat teknologi IoT di *Greenhouse* diperlukan sensor yang dapat membaca lingkungan dari *greenhouse* dan juga jumlah nutrisi yang diberikan, setiap sensor perlu terhubung dengan internet dan pada bagian kendali pun perlu sebuah *Aktuator* yang terhubung di internet sehingga otomasi dapat dilakukan dengan menggunakan penjadwalan ataupun otomatis berdasarkan pembacaan sensor.

Pada teknologi IoT dalam melakukan monitoring diperlukan sebuah media yang dapat pengguna gunakan untuk menampilkan data-data monitoring pada *greenhouse* [10]. Menurut penelitian yang dilakukan oleh Erlangga Prasetya et al data-data sensor IoT dapat dikirim dan

dilihat di dalam website [11]. Selain itu menurut penelitian yang dilakukan oleh Hardani el et android dapat menjadi media untuk melakukan monitoring dan kendali IoT [12]. Dari android dan juga web dapat menjadi media untuk pengguna memudahkan melakukan monitoring IoT yang ada di *greenhouse*, akan tetapi website dapat dibuka pada semua platform dengan menggunakan *browser* sehingga website cocok untuk pengembangan awal dari sistem IoT dikarenakan akses website menjadi pilihan dalam penelitian ini untuk menampilkan data-data monitoring dan data-data sensor IoT [13].

Untuk menggambarkan sebuah website Terdapat alur ataupun tahapan-tahapan dalam pembuatan sebuah perangkat lunak, sehingga proses pembuatan website menjadi lebih mudah dan terarah atau biasa dikenal sebagai metode pengembangan perangkat lunak atau *Software Development Life Cycle* (SDLC). Terdapat berbagai macam metode penembangan perangkat lunak yang biasanya digunakan berdasarkan riset yang dilakukan oleh Gagan Gurung el at Model yang menjadi awal dari metode pengembangan perangkat lunak adalah metode Waterfall terdapat kelebihan dari metode ini adalah cocok digunakan dalam proyek yang kecil dan memiliki kejelasan dalam fitur, akan tetapi waterfall memiliki kelemahan, sulit untuk mengetahui waktu pengembangan, tidak cocok untuk proyek yang kompleks. Lalu pada metode Iterative memiliki keunggulan yakni waktu pengembangan yang dapat diukur, Tiap fungsi dapat dibuat dengan cepat, testing dan debugging lebih mudah dikontrol, akan tetapi memiliki kelemahan dalam proyek yang kecil, membutuhkan tenaga kerja yang lebih banyak dan pada pembuatan requirement tidak semua dapat ditulis. Spiral memiliki kelebihan dalam skalabilitas dan juga dalam pembuatan perangkat lunak memiliki penangan risiko yang baik, kekurangan tidak cocok dengan tim kecil, dan membutuhkan tenaga kerja yang profesional. V shape, memiliki kelebihan dalam testing dan validasi di setiap tahapan, memiliki kelemahan dalam pembuatan proyek yang kecil dan tidak fleksibel. Agile memiliki kelebihan yakni penanganan risiko yang baik, cepat untuk mengembangkan perangkat lunak, dan bagus untuk proyek yang memiliki kebutuhan yang mudah berubah ubah. Memiliki kelemahan yakni agile metode yang berfokus pada klien, sulit untuk membuat sistem yang permanen, proyek dapat melebihi batas pengembangan karena fitur yang belum jelas [14]. Sehingga berdasarkan kelebihan dan kekurangan yang ada metode agile merupakan metode yang cocok digunakan dalam pengembangan sistem ini dikarenakan memiliki klien sehingga fitur bisa saja berubah jika tidak sesuai dari kebutuhan klien.

Setelah memilih metode pengembangan agile, agile terdapat banyak *framework* seperti *scrum*, *extreme programming*, *Scaled Agile Framework*, *Lean Software Development*,

Crystal Methodology, Dynamic Systems Development Method, dan Feature Driven Development (FDD). Dari semua metode *agile* metode-metode tersebut banyak memiliki kelebihan yakni jika sebuah aplikasi dibuat secara berkelompok akan tetapi dikarenakan penelitian ini dibuat tidak menggunakan team maka metode yang cocok adalah *personal extreme programming*, dan juga *kanban* dikarenakan kedua metode ini tidak menekankan pada kerja secara tim[15]. Metode kanban dapat dilakukan dan memiliki kelebihan yakni pembuatan menjadi terarah dan dapat mengetahui hambatan yang terjadi saat mengerjakan sebuah fitur *website* [16]. Menurut penelitian yang dilakukan oleh Parman Suparman dan Miftachul Huda penerapan Metode Kanban dalam pengembangan perangkat lunak mampu membuat tugas-tugas pengembangan lebih jelas dan mempunyai alur terpadu, sehingga mampu mempercepat pengembangan dan menekan biaya produksi. Sehingga pada penelitian ini akan menggunakan metode pengembangan *agile kanban* [17].

Metode pengembangan perangkat lunak memiliki bagian pengujian yang diperlukan untuk menguji sistem dari error dan juga bug. Terdapat pengujian yang sering digunakan yakni blackbox testing dan juga whitebox testing. Pada penelitian ini menggunakan metode black-box dikarenakan yakni white box dibutuhkan pembuatan unit test yang digunakan dalam website, dikarenakan pengembangan tidak hanya berfokus pada *website* tetapi IoT maka untuk menguji sistem berhasil membaca sensor dan aktuator menggunakan *black-box testing* dikarenakan dengan *black-box testing* dapat mengetahui kesalahan yang tidak terdeteksi secara kode tetapi secara fungsionalitas tidak berjalan sesuai yang diinginkan [18].

Selain pengujian kode dikarenakan sistem dibuat untuk klien diperlukan pengujian apakah sistem sudah sesuai dengan kebutuhan dari klien. Dalam pengujian ini biasanya terdapat banyak metode yang dapat digunakan dalam menguji sistem terhadap kebutuhan user seperti SUS (*System Usability Scale*) yang merupakan pengujian yang dapat dilakukan secara cepat dan valid[19]. Sehingga, pada penelitian ini melakukan evaluasi sistem menggunakan *System Usability Scale* dikarenakan waktu evaluasi dan partisipan yang digunakan tidak perlu banyak dan penelitian ini berfokus pada klien dan akan digunakan oleh pemilik dari *greenhouse* ibu Zunanik Mufidah, S.TP., M.Si ataupun penjaga dari *greenhouse* sehingga untuk mengetahui apakah sistem dapat dengan mudah ataupun tidak terjadi masalah saat digunakan untuk melakukan monitoring dan kontroling pada *greenhouse* maka diperlukan pengujian berupa *System Usability Scale* atau SUS[20].

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, adapun rumusan masalah yang diambil oleh penulis yakni

1. Bagaimana membangun sistem monitoring smart *greenhouse* berbasis IoT menggunakan metode *agile kanban*?
2. Bagaimana Hasil evaluasi dari sistem monitoring smart *greenhouse* berbasis IoT dengan menggunakan metode *System Usability Scale* (SUS) dan *Black box Testing*?
3. Bagaimana hasil perbandingan dari penanaman melon menggunakan sistem secara konvensional dan secara smart *greenhouse* berbasis IoT?

1.3 Tujuan Penelitian

Tujuan dari penelitian ini adalah

1. Menghasilkan sistem monitoring smart *greenhouse* berbasis IoT menggunakan metode *agile kanban*.
2. Mengetahui hasil evaluasi sistem monitoring smart *greenhouse* berbasis IoT dengan menggunakan metode *System Usability Scale* (SUS) dan *Black box Testing*.
3. Mengetahui hasil dari perbandingan penanaman melon menggunakan sistem secara konvensional dan secara smart *greenhouse* berbasis IoT.

1.4 Batasan Masalah

Batasan masalah dalam penelitian ini diperlukan agar menghindari adanya penyimpangan maupun pelebaran pokok masalah agar penelitian tersebut lebih terarah dan memudahkan dalam pembahasan sehingga tujuan penelitian akan tercapai. Beberapa batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Pembacaan sistem debit hanya membaca jika dialiri nutrisi.
2. Data yang ditampilkan sistem yang dibangun berupa aplikasi berbasis web dan sistem hanya dapat dibuka melalui browser,
3. Web hanya dapat dibuka jika pengguna terhubung dengan internet dan juga alat dapat membaca jika terhubung internet.
4. Sistem hanya melakukan monitoring pada suhu, kelembaban, debit sehingga Batasan hanya pada lingkungan dan tidak pada masalah melon lainnya.

5. Sistem hanya melakukan kendali pada kipas, *cooling pad*, pompa nutrisi, jumlah nutrisi
6. Penelitian ini dilakukan pada *greenhouse* ukuran 25m x 12.5m.
7. Sistem memiliki power supply berupa tegangan listrik akan tetapi tidak akan dibahas rancangan dari power supply dari sistem IoT.

1.5 Manfaat Penelitian

Manfaat yang akan didapatkan dari penelitian ini yaitu sebagai berikut :

1. Dengan adanya sistem ini dapat melakukan monitoring *greenhouse* mulai dari mengatur suhu, kelembaban, dan juga pemberian nutrisi. Sehingga dengan adanya sistem monitoring ini dapat meningkatkan hasil panen melon lebih baik mulai dari bobot, ukuran melon dan tingkat kemanisan dari melon.
2. Hasil dari sistem dapat menjadi salah satu inovasi yang dapat dikembangkan kedepanya oleh penelitian lainnya sehingga didapatkan sistem yang lebih baik dan dapat digunakan tidak hanya di Greenhouse Institut Teknologi Sumatera.

1.6 Sistematika Penulisan

Sistematika dalam penelitian ini yaitu:

1.6.1 Bab I Pendahuluan

Pada bab ini menjelaskan latar belakang, identifikasi masalah, rumusan masalah, tujuan penelitian, batasan masalah, dan sistematika penulisan dari penelitian yang akan dilakukan.

1.6.2 Bab II Tinjauan Pustaka

Pada bab ini berisi tinjauan pustaka yang terdiri dari tinjauan pustaka dari penelitian-penelitian yang relevan dan tinjauan studi yang menjelaskan teori-teori dasar penelitian yang mendukung penelitian ini.

1.6.3 Bab III Metodologi Penelitian

Pada Bab ini berisi mengenai metode penelitian seperti alur penelitian, penjabaran langkah penelitian, alat dan bahan, metode pengembangan, dan rancangan pengujian.

1.6.4 Bab IV Hasil dan Pembahasan

Pada Bab ini berisi penjabaran mengenai implementasi penelitian berdasarkan rancangan metode penelitian yang ditetapkan sebelumnya serta hasil pengujian dari sistem yang telah dibangun.

1.6.5 Bab V Kesimpulan dan Saran

Bab ini berisi penyajian hasil penelitian dan analisa dari hasil penelitian kesimpulan serta saran untuk penelitian

BAB II

LANDASAN TEORI

3.1 Tinjauan Pustaka

Pada penelitian ini, peneliti mendapatkan tinjauan pustaka dari penelitian-penelitian yang berkaitan dengan pengembangan IoT pada *greenhouse*. dari beberapa tinjauan tersebut tentunya dapat menjadi bahan untuk melakukan perbandingan dan pertimbangan dalam mendukung pembuatan penelitian ini. Penelitian-penelitian yang berkaitan dengan pengembangan IoT pada *greenhouse* dapat dilihat pada Tabel 2.1.

Penelitian pertama yang dilakukan oleh Agha Pradipta Merdekawan , Putriana Sari pada tahun 2020 dengan judul Studi Awal Rancang Bangun Indoor Farming Monitoring System Berbasis IoT dengan Protokol Websocket berhasil membuat sebuah sistem yang dapat mengirimkan data-data sensor yang dapat dibaca pada android menggunakan protokol websocket dan menggunakan metode waterfall. jaringan yang digunakan berupa mqtt dan komunikasi broker dan aplikasi menggunakan websocket berhasil dilakukan sehingga dapat disimpulkan protokol mqtt merupakan protokol yang dapat mengirimkan data-data sensor untuk melakukan monitoring serta controlling pada *smart farming* [21].

Penelitian kedua oleh Uray Ristian, Ikhwan Ruslianto, Kartika Sari pada tahun 2022 yang berjudul Sistem Monitoring Smart *Greenhouse* pada Lahan Terbatas Berbasis Internet of Things (IoT) Sistem berhasil dibuat dengan berhasil membaca kondisi Kelembaban Tanah, Kelembaban Udara, Suhu dan pH Tanah dan data tersebut dikirimkan ke server untuk ditampilkan ke Web Browser pada PC Sehingga dapat digunakan untuk melakukan monitoring pada *greenhouse*[22].

Penelitian ketiga oleh Muhammad Shoaib Farooq, Rizwan Javid, Shamyla Riaz, And Zabihullah Ata pada tahun 2022 yang berjudul IoT Based Smart *Greenhouse* Framework and Control Strategies for Sustainable Agriculture. Pada penelitian ini *greenhouse* memiliki monitoring dan kontrol yang dapat mengendalikan hal-hal yang sebelumnya jadi masalah yakni tenaga kerja, nutrisi, ataupun iklim dari tanaman dengan menggunakan sensor dan Aktuator[23].

Penelitian keempat oleh B. Lanitha, E. Poornima, R. Sudha, D. Beulah David, K. Kannan, R. Jegan, Vijayakumar Peroumal ,R. Kirubagharan, Meroda Tesfaye pada tahun 2022 yang berjudul IoT Enabled Sustainable Automated *Greenhouse* Architecture with

Machine Learning Module. Pada penelitian ini dibuat sebuah sistem IoT yang dapat mengendalikan lingkungan dari sebuah sensor suhu, kelembaban yang dapat mengatur kipas, serta dibuatnya sistem pemberian nutrisi berdasarkan kelembaban kondisi dari tanah dengan menggunakan machine learning untuk prediksi kebutuhan pupuk. Sistem terhubung dengan perangkat keras menggunakan blynk [24].

Penelitian kelima oleh Muhammad Yanuar Muhammin, Aulia Rahma Annisa, Billy Montolalu pada tahun 2022 yang berjudul Rancang Bangun Smart System Green House untuk Budidaya Melon Berbasis PLC. Hasil dari penelitian ini adalah pertumbuhan tanaman melon pada *greenhouse* berbasis PLC menunjukkan hasil yang lebih baik dibandingkan dengan tanpa menggunakan PLC. Hal tersebut terlihat pada perbedaan pertumbuhan daun, berat buah, pertumbuhan bunga dan kelembaban tanah. Monitoring kadar nutrisi tanaman telah berhasil 90% sesuai dengan ketentuan nutrisi pertumbuhan tanaman melon [25].

Tabel 2.1 Penelitian Terdahulu

| No | Nama Penulis dan Tahun | Judul | Masalah | Metode | Hasil | Keunggulan atau Keterbatasan |
|----|--|---|---|-----------|---|---|
| 1 | Agha Pradipta Merdekawan, Putriana Sari (2022) | Studi Awal Rancang Bangun Indoor Farming Monitoring System Berbasis IoT dengan Protokol Websocket | Menurunnya kapasitas lahan pertanian, jumlah lahan pertanian, degradasi lahan dan air, serta konversi lahan pertanian | Waterfall | Membuat sistem monitoring perkebunan dalam ruangan. Dengan IoT sistem dapat memonitoring dan controlling tanaman hidroponik dengan menggunakan aplikasi Android dengan menggunakan Protokol WebSocket | Sistem belum melakukan kontrol otomatis misalnya pada penyiraman otomatis, ataupun menyalaikan lampu LED UV otomatis. |

| No | Nama Penulis dan Tahun | Judul | Masalah | Metode | Hasil | Keunggulan atau Keterbatasan |
|----|---|---|--|------------------------------------|---|--|
| No | Nama | Judul | Masalah | Metode | Hasil | Keunggulan |
| 2 | Uray Ristian, Ikhwan Ruslianto, Kartika Sari (2022) | Sistem Monitoring Smart <i>Greenhouse</i> pada Lahan Terbatas Berbasis Internet of Things (IoT) | Beberapa faktor lingkungan yang sangat berpengaruh pada proses pertumbuhan dan kesuburan tanaman adalah Faktor suhu, air, kelembaban tanah, kelembaban udara, dan cahaya. | - | Membuat sistem Sistem Monitoring Smart <i>Greenhouse</i> berhasil membaca kondisi Kelembaban Tanah, Kelembaban Udara, Suhu dan pH Tanah dan data tersebut dikirimkan ke server untuk ditampilkan ke Web Browser pada PC | Tidak seperti penelitian pertama yang melakukan pengolahan pada data menjadi grafik pada penelitian ini tidak terdapat penjadwalan otomatis dari Aktuator. |
| 3 | Muhammad Shoaib Farooq, Rizwan Javid, Shamsya Riaz, And Zabihullah Ata.(2022) | IoT Based Smart <i>Greenhouse</i> Framework and Control Strategies for Sustainable Agriculture | Kebutuhan hasil pertanian untuk memenuhi kebutuhan pangan semakin dibutuhkan terdapat masalah pada pertanian mulai dari tenaga kerja, pemberian nutrisi, hama, dan perubahan iklim yang ekstrim. | systematic literature review(SLR) | Pada <i>greenhouse</i> memiliki monitoring yang dapat mengendalikan hal-hal yang sebelumnya jadi masalah yakni tenaga kerja, dan iklim dari tanaman | Terdapat permasalahan dalam sistem IoT yakni rawan akan keamanan dari sistem mulai bocornya data interupsi dari luar. |

| | Penulis dan Tahun | | | | | atau Keterbatasan |
|----|--|--|--|--------|--|---|
| No | Nama Penulis dan Tahun | Judul | Masalah | Metode | Hasil | Keunggulan atau Keterbatasan |
| 4 | B. Lanitha, E. Poornima, R. Sudha, D. Beulah David, K. Kannan, R. Jegan, Vijayakumar Peroumal, R. Kirubaghara ran, Meroda Tesfaye (2022) | IoT Enabled Sustainable Agriculture with Machine Learning Module | Efisiensi dan hasil dari pertanian didasarkan dari permasalahan klimatologi dan juga masalah dari pemberian nutrisi. Dibutuhkan sebuah sistem yang dapat memberikan efisiensi dalam melakukan perawatan ataupun manajemen dari pemberian nutrisi dan pengontrol kondisi lingkungan | - | Dibuat sebuah sistem IoT yang dapat mengendalikan lingkungan dari sebuah sensor suhu, kelembaban yang dapat mengatur kipas, serta dibuatnya sistem pemberian nutrisi berdasarkan kelembaban kondisi dari tanah dengan menggunakan machine learning untuk prediksi kebutuhan pupuk. Sistem terhubung dengan perangkat keras menggunakan blynk | Sistem yang dibuat memiliki kelebihan yakni adanya sistem cerdas yang dapat memberikan prediksi untuk kebutuhan nutrisi |

| | | | | | | |
|---|--|---|--|---|--|---|
| 5 | Muhammad Yanuar Muhaimin, Aulia Rahma Annisa, Billy Montolalu (2022) | Rancang Bangun Smart System Green House untuk Budidaya Melon Berbasis PLC | Melon merupakan tanaman yang tumbuh dipengaruhi oleh cuaca dan jumlah nutrisi yang diberikan. Jika tidak sesuai dapat berpengaruh pada melon yang tumbuh dengan ukuran yang kecil dan rasa yang kurang manis | - | Hasil pertumbuhan tanaman melon pada <i>greenhouse</i> berbasis PLC menunjukkan hasil yang lebih baik dibandingkan dengan tanpa menggunakan PLC. Hal tersebut terlihat pada perbedaan pertumbuhan daun, berat buah, pertumbuhan bunga dan kelembaban tanah. Monitoring kadar nutrisi tanaman telah berhasil 90% sesuai dengan ketentuan nutrisi pertumbuhan tanaman melon. | Sistem PLC menjadi sistem yang dapat melakukan otomatisasi penjadwalan dan juga manual secara baik dan lebih cepat untuk dikembangkan akan tetapi sistem PLC memiliki harga relatif lebih mahal dan perlu memiliki sumber tegangan yang lebih besar dibandingkan menggunakan mikrokontroler |
|---|--|---|--|---|--|---|

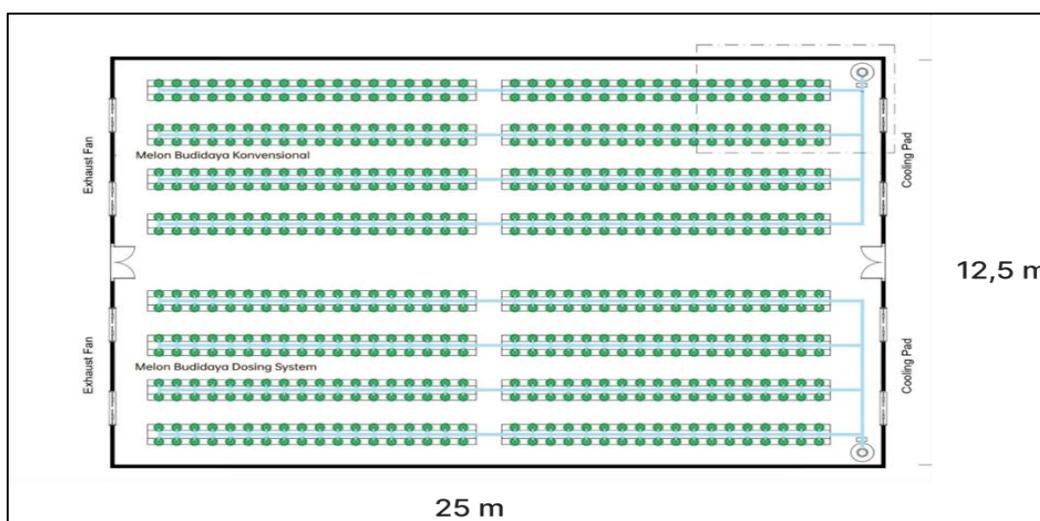
Berdasarkan penelitian terdahulu dapat disimpulkan sistem perlu dibuat dapat melakukan monitoring pada kendali lingkungan, pemberian nutrisi dan juga kendali diperlukan sistem otomatisasi seperti penjadwalan dan juga otomatis berdasarkan sensor, penggunaan dilakukan dengan menggunakan mikrokontroler yang memiliki harga yang lebih terjangkau dibandingkan sistem PLC. Menampilkan data monitoring yang dapat dilakukan dengan menggunakan website dan diperlukan sebuah protokol jaringan untuk dapat menghubungkan antara website dengan data sensor serta database yang diperlukan untuk menyimpan data-data untuk riwayat data sensor ataupun sistem kendali.

3.2 Dasar Teori

Berisi teori maupun konsep yang berkaitan yang digunakan dalam tugas akhir berikut adalah dasar teori yang digunakan dalam penelitian ini :

2.2.1 Smart Greenhouse

Greenhouse adalah sebuah lingkungan untuk tanaman dapat tumbuh dengan kondisi yang dapat dikendalikan. *Greenhouse* dibuat untuk budidaya hortikultura dikarenakan penggunaannya sangat penting sebagai penjamin keberhasilan tumbuh dari pengaruh lingkungan seperti suhu, kelembaban udara, intensitas matahari, dan hama penyakit. Penggunaan *greenhouse* dalam budidaya tanaman merupakan salah satu cara untuk memberikan lingkungan yang dapat memberikan tanaman pada kondisi optimum bagi pertumbuhan tanaman. *Greenhouse* dikembangkan pertama kali dan umum digunakan di kawasan yang beriklim subtropika. Penggunaan *greenhouse* terutama ditujukan untuk melindungi tanaman dari suhu udara yang terlalu rendah pada musim dingin. Menurut Nelson *greenhouse* merupakan suatu bangunan untuk budidaya tanaman, yang memiliki struktur atap dan dinding yang bersifat tembus cahaya. Cahaya masuk dengan tambahan plastic uv dapat masuk ke dalam *greenhouse* dan memenuhi kebutuhan dari Cahaya dan kondisi lainnya dapat terkendali seperti suhu udara menghindari curah hujan yang terlalu tinggi, dan tiupan angin yang terlalu kencang[26]. Pada *greenhouse* itera memiliki ukuran 25m x 12,5 m x 3m dengan total jumlah tanaman melon sebanyak 361 buah melon seperti pada gambar 2.1 .



Gambar 2.1 Ilustrasi *Greenhouse* ITERA

Greenhouse yang digunakan dengan menggunakan tiang yang terbuat dari besi sehingga dapat menyangga secara tahan lama dan bahan plastik UV dengan tebal 150 mikron untuk menutupi *greenhouse* hal ini dikarenakan menurut penelitian yang dilakukan oleh

Rusdi Faizin, Dedy Darmansyah, dan Yeti Darnila bahwa UV diklaim kuat terhadap iklim dan juga UV dapat mengendalikan kondisi suhu di *greenhouse* yang akan relatif stabil dan normal, tidak bakal merusak tanaman, pengeringan secara merata juga maksimal, sehingga tanaman bisa bertahan lama dan tidak gampang ditumbuhi jamur[27]. Sehingga dengan plastik UV juga dapat melindungi sensor dan alat elektronik lainnya dari hujan ataupun air berikut adalah gambaran dari *greenhouse* ITERA yang menggunakan plastik UV pada gambar 2.2 berikut ini.



Gambar 2.2 Tampak depan dari *Greenhouse* ITERA

Namun banyak *greenhouse* yang cara pengontrolannya masih secara manual yaitu dengan menyiram tanaman dan pengontrolan setiap hari yang dilakukan oleh manusia hal ini menjadi masalah karena tanaman melon menjadi salah satu buah yang perlu memiliki banyak tenaga kerja dalam perawatanya. Dengan berkembangnya teknologi Internet of Things (IoT) dapat diterapkan pada *greenhouse* sehingga pengontrolan dapat dibuat menjadi otomatis dan dapat dimonitoring secara jarak jauh. Dalam hal ini penelitian berfokus pada sistem pengontrolan *greenhouse* yang berbasis teknologi Internet of Things (IoT) dan mikrokontroler untuk menjalankan semua perintah yang diinginkan untuk mendukung smart green house. Adapun kelebihan IoT itu sendiri adalah dapat menghubungkan pengguna lebih mudah berinteraksi dengan semua peralatan yang terhubung dengan internet [28].

2.2.2 Sistem Monitoring

Menurut Kamus Besar Bahasa Indonesia (KBBI), kata "monitoring" berasal dari kata "monitor", yang berarti "orang yang memantau atau mengawasi". Monitoring dapat didefinisikan dalam pedoman ini sebagai sebuah kegiatan pengamatan yang bertujuan untuk

memberikan informasi tentang sebab dan akibat dari kebijakan baru atau proses yang lebih terfokus pada kegiatan yang dilakukan selama satu tahun. Monitoring dilakukan untuk memastikan bahwa kegiatan berlangsung sesuai dengan perencanaan dan prosedur yang telah disepakati. Secara umum, monitoring dilakukan selama kegiatan berlangsung untuk memastikan apakah proses dan pencapaian sesuai dengan rencana. Kegiatan dapat berjalan sesuai rencana dengan memperbaiki masalah segera setelah ditemukan. Hasil observasi menentukan pentingnya proses selanjutnya. Indikator yang digunakan dalam kegiatan pemantauan mencakup penting Kegiatan dapat berjalan sesuai rencana dengan memperbaiki masalah segera setelah ditemukan. Hasil observasi menentukan pentingnya proses selanjutnya. Indikator untuk kegiatan monitoring termasuk pentingnya kegiatan dan tujuan yang telah ditetapkan dalam perencanaan kegiatan. Apabila monitoring dilakukan dengan benar, itu akan membantu memastikan bahwa kegiatan dilakukan sesuai dengan pedoman dan perencanaan. Selain itu, indikator membantu pengelola dalam melakukan evaluasi dan memberikan informasi apabila terjadi hambatan atau penyimpangan. [29]. Sistem monitoring akan mempermudah suatu pekerjaan jika dirancang dan dilakukan secara efektif. Dalam sistem ini yang dimonitoring adalah kondisi ruangan dan nutrisi yang diberikan.

2.2.3 Internet of Things

Internet of Things, atau singkatnya IoT, adalah sebuah konsep yang bertujuan untuk memperluas manfaat dari koneksi internet yang selalu aktif yang memungkinkan kita menghubungkan mesin, perangkat, dan objek fisik lainnya melalui sensor dan aktuator online untuk mendapatkan informasi dan mengelolanya. pertunjukan . sehingga mesin dapat bekerja sama dan bahkan bertindak secara mandiri berdasarkan informasi yang baru diperoleh. Internet of Things, atau Internet of Things seperti yang sering disebut, adalah sebuah ide di mana semua objek di dunia nyata dapat berkomunikasi satu sama lain sebagai bagian dari sistem yang terintegrasi, menggunakan Internet sebagai penghubung. Misalnya, CCTV yang dipasang di pinggir jalan terhubung ke Internet dan terhubung ke ruang kontrol yang jaraknya bisa puluhan kilometer. atau ke smart home yang bisa dikendalikan melalui koneksi internet. Pada dasarnya perangkat IoT terdiri dari sensor sebagai alat pengumpul data, koneksi internet sebagai alat komunikasi, dan server sebagai pengumpul data untuk menerima dan menganalisis sensor. Kevin Ashton pertama kali mempresentasikan ide Internet of Things pada tahun 1999 dalam salah satu presentasinya. Kini banyak perusahaan besar seperti Intel, Microsoft, Oracle dan masih banyak lainnya yang mulai merambah IoT. Banyak

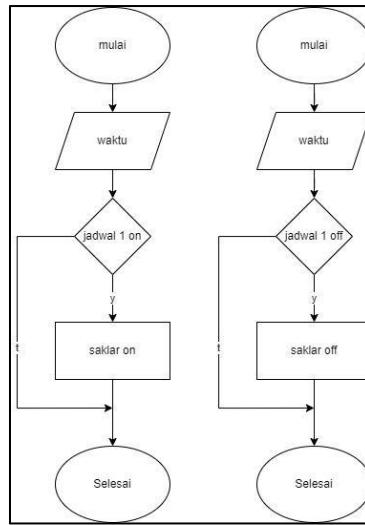
yang memperkirakan bahwa dampak Internet of Things akan menjadi “next big thing” dalam dunia teknologi informasi, karena Internet of Things menawarkan banyak potensi yang dapat digali. Contoh sederhana manfaat dan penerapan Internet of Things misalnya kulkas yang dapat menginformasikan pemilik melalui SMS atau email. makanan dan minuman mana yang sudah habis dan perlu diisi ulang.[30].

2.2.4 Otomatisasi dan Penjadwalan

jadwal sendiri berdasarkan KBBI merupakan pembagian waktu berdasarkan rencana pengaturan urutan kerja; daftar atau tabel kegiatan atau rencana kegiatan dengan pembagian waktu pelaksanaan yang terperinci. Sehingga dapat dikatakan otomatisasi merupakan pembagian waktu yang perlu dilakukan dalam penyiraman. Berdasarkan observasi yang dilakukan di *Greenhouse ITERA*, Tanaman melon diberikan penyiraman 2x sehari dengan menggunakan nutrisi AB Mix dengan 1600 ppm. Untuk hasil yang maksimal menurut penelitian yang dilakukan oleh santri dan agus pada pengaruh interval waktu pemberian Nutrisi Ab-Mix dan Metode Hidroponik yakni pada pada interval 2 jam (07.00, 09.00, 11.00, 13.00, 15.00) Sehingga didapatkan waktu untuk memberikan nutrisi sebanyak 5x dalam 1 hari, hal ini dikarenakan pemberian interval 2 jam sekali dapat meningkatkan bobot buah sebesar 22,68% pada buah melon [31]. Lalu setelah adanya jadwal diperlukannya otomatisasi, menurut kbbi otomatisasi adalah proses, cara, perbuatan menjadwalkan atau memasukkan dalam jadwal. Atau dapat diartikan sebagai sebuah pembuatan jadwal untuk dilaksanakan dalam penelitian ini otomatisasi dilakukan oleh sistem. Pada riset yang dilakukan oleh Agha Terdapat 3 cara yang dapat dilakukan dalam otomatisasi sistem IoT yakni :

2.1 Klasik

metode klasik, yaitu metode penjadwalan yang menggunakan waktu untuk merubah posisi saklar on dan off dengan membuat setiap jadwal waktunya seperti Seperti pada gambar 2.3.

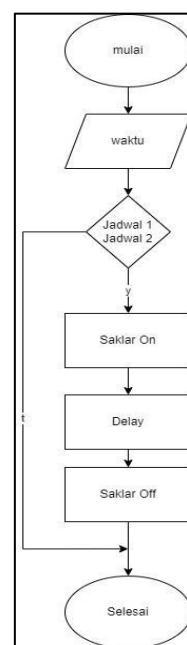


Gambar 2.3 Penjadwalan metode klasik

Sumber gambar : Smart Garden Automation Dengan Memanfaatkan Teknologi Berbasis Internet Of Things (IoT) [32]

2.2 Delay Time

Metode Automation Delay Time (Penundaan Waktu Otomatis), yaitu metode penjadwalan yang sudah menggunakan algoritma keputusan if .. then dengan waktu sebagai masukan kondisi dan aksi saklar On ke Off menggunakan penundaan waktu. Seperti pada gambar 2.4.

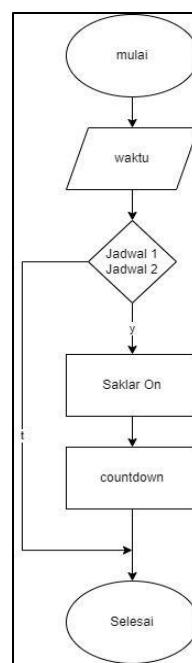


Gambar 2.4 Penjadwalan metode Delay Time

Sumber Gambar : Smart Garden Automation Dengan Memanfaatkan Teknologi Berbasis Internet Of Things (IoT) [32]

2.3 Countdown

Metode Automation Countdown (Perhitungan Mundur Otomatis), yaitu metode penjadwalan yang sudah menggunakan algoritma keputusan if .. then dengan waktu sebagai masukan kondisi dan aksi saklar On ke Off menggunakan perhitungan mundur Seperti pada gambar 2.5 [32].

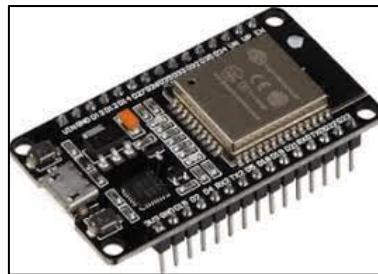


Gambar 2.5 Penjadwalan metode Countdown

Sumber Gambar : Smart Garden Automation Dengan Memanfaatkan Teknologi Berbasis Internet Of Things (IoT) [32]

2.2.5 Mikrokontroller

Mikrokontroler merupakan sebuah komputer mikro memiliki tiga komponen utama, yaitu: unit pengolahan pusat (CPU: Central Processing Unit), memori dan system I/O (Input/output) untuk dihubungkan ke perangkat luar. CPU yang mengatur sistem kerja komputer mikro, dibangun oleh sebuah mikroprosesor. Memori terdiri atas GEPROM untuk menyimpan program dan RAM untuk menyimpan data. Sistem I/O bisa dihubungkan dengan perangkat luar misalnya sebuah keyboard dan sebuah monitor, bergantung pada aplikasinya. Apabila CPU, memori dan sistem I/O dalam sebuah chip semikonduktor, maka inilah yang dinamakan mikrokontroler [33]. Terdapat mikrokontroler yang dapat terhubung dengan internet menggunakan module wifi salah satunya adalah esp8266 dan esp 32 seperti gambar 2.5 sebagai berikut :



Gambar 2.6 Esp-8266

Sumber gambar : https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcS4BuCBAvJ78u4-gXaq_19gdEywSpgRl5Sycg&usqp=CAU

Mikrokontroler yang digunakan merupakan esp8266 dengan datasheet pada tabel 2.2.

Tabel 2.2 Datasheet ESP8266

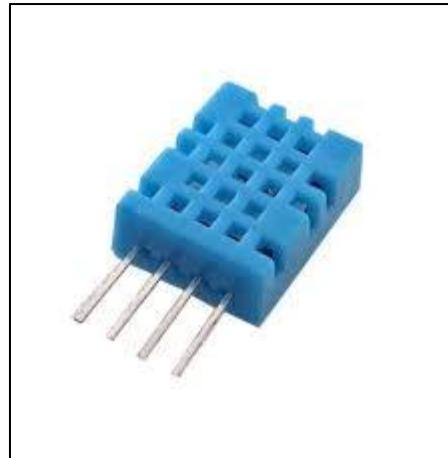
| no | Parameter | Keterangan |
|----|------------------------|------------|
| 1 | Tegangan input | 7-12V |
| 2 | Tegangan Operasi | 3.3V |
| 3 | Pin Digital I/O (DIO) | 16 |
| 4 | Pin Analog Input (ADC) | 1 |
| 5 | Flash Memory | 4 MB |
| 6 | SRAM | 64 KB |
| 7 | Clock Speed | 80 MHz |
| 8 | Wifi | Ada |

2.2.6 Sensor

Sensor adalah jenis transduser yang digunakan untuk mengubah besaran mekanis, magnetis, panas, sinar dan kimia menjadi tegangan dan arus listrik. Sensor biasa digunakan untuk pendektsian pada saat melakukan pengukuran dan pengendalian. Beberapa jenis sensor yang banyak digunakan dalam rangkaian elektronik antara lain adalah sensor cahaya, sensor suhu dan sensor tekanan[34]. Pada penelitian ini sensor yang digunakan adalah sensor suhu, kelembaban, intensitas cahaya, debit nutrisi, ph nutrisi, suhu nutrisi, kepekatan nutrisi.

1. Sensor Suhu & Kelembaban

Sensor suhu yang digunakan adalah sensor dht 11



Gambar 2.7 Sensor DHT 11

Sumber Gambar : <https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTnPT6sklwqr9YKlgYDmSXanYdbfflU1CPuQg&usqp=CAU>

Datahseet dari sensor dht 11 dapat dilihat pada tabel 2.3

Tabel 2.3 Datasheet DHT 11

| no | Parameter | Keterangan |
|----|------------------|------------|
| 1 | Range Suhu | 0-50 C |
| 2 | Range Kelembaban | 20-90%RH |
| 3 | Jumlah pin | 4 |
| 4 | Power Supply | DC |
| 5 | Tegangan input | 3V - 5.5v |

2. Sensor Debit



Gambar 2.8 Sensor Debit

Sumber Gambar :

https://s1.bukalapak.com/img/157987125/large/G1_2_Water_Flow_Sensor__Sensor_Debit_Air__Flow_Meter_Digit.jpg

Datasheet dari sensor debit dapat dilihat pada tabel 2.4.

Tabel 2.4 Datasheet Sensor Debit

| no | Parameter | Keterangan |
|----|-----------------|----------------|
| 1 | Range Pembacaan | 0 - 30 L |
| 2 | Power Supply | DC |
| 3 | Tegangan input | 4.5V to 18V DC |

2.2.7 Aktuator

Aktuator adalah sebuah alat mekanis yang mengubah tenaga listrik maupun fluida menjadi kuantitas lain seperti kecepatan dan perangkat elektromagnetik sehingga mampu menghasilkan energi kinetik. Energi kinetik yang dihasilkan akan digunakan untuk menggerakkan atau mengontrol sebuah mekanisme atau sistem.

Biasanya Aktuator diaktifkan oleh lengan mekanik yang digerakkan oleh motor listrik. Alat mekanis ini dikendalikan oleh pengontrol otomatis yang telah diprogram di antara mikrokontroler. Aktuator sendiri dapat melakukan hal-hal tertentu setelah menerima perintah dari controller. Salah satu bentuk Aktuator adalah pompa dan relay yang akan digunakan dalam penelitian ini [35].

1. Selenoid Valve

Solenoid valve untuk penyiraman pada tanaman



Gambar 2.9 Selenoid Valve

Sumber Gambar : [batch-upload_0a61b977-e804-44cb-a8b8-3c2a4816bb52.jpeg \(700x700\) \(tokopedia.net\)](batch-upload_0a61b977-e804-44cb-a8b8-3c2a4816bb52.jpeg (700x700) (tokopedia.net))

dengan datasheet

Tabel 2.5 Datasheet Selenoid Valve

| no | Parameter | Keterangan |
|----|----------------|------------|
| 1 | Ukuran | 1 Inch |
| 2 | Power Supply | AC |
| 3 | Tegangan input | 220V |

2. Relay

Relay yang digunakan merupakan relay optocoupler 5v



Gambar 2.10 Relay

Sumber Gambar : <https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTLWqQTTrmF0bS1tEK2dUYN2U9NzADDdFZY7A&usqp=CAU>

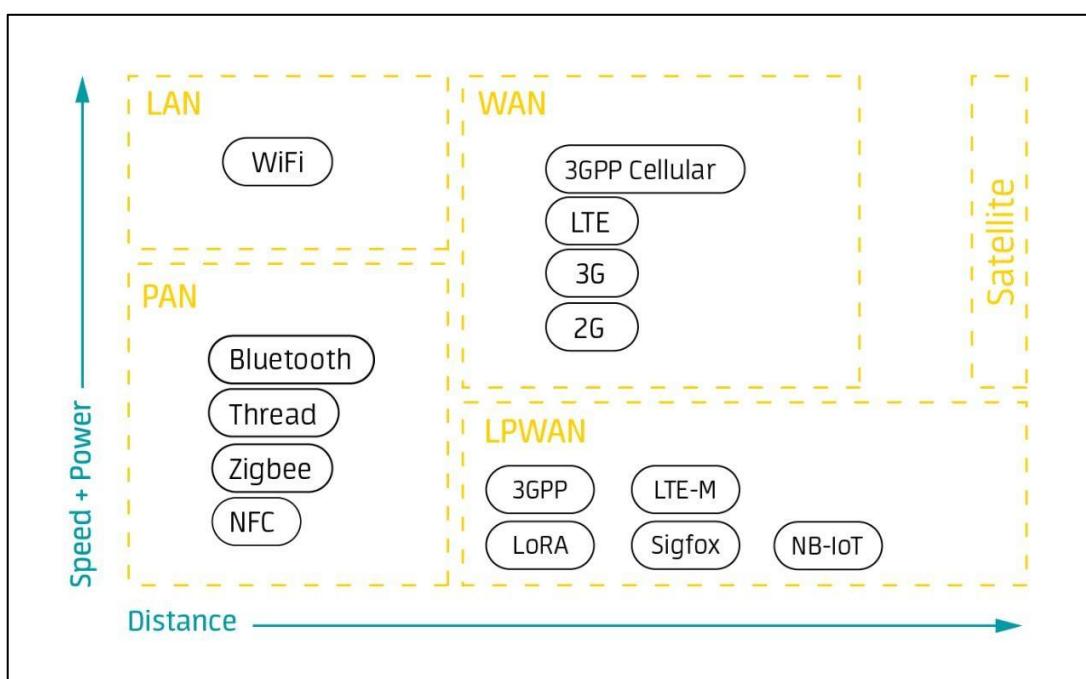
Terdapat datasheet dari solenoid valve yang dapat dilihat detailnya pada tabel 2.6

Tabel 2.6 Datasheet Selenoid Valve

| no | Parameter | Keterangan |
|----|----------------------|-------------------|
| 1 | Tegangan Output | 250VAC atau 30VDC |
| 2 | Arus Output maksimum | 10A |
| 3 | Power Supply | DC |
| 4 | Tegangan Input | 3.75V - 6V |

2.2.8 Jaringan Komunikasi

Dalam menggunakan IoT diperlukan jaringan komunikasi antara alat dengan sebuah perangkat, untuk dapat saling berkomunikasi diperlukan sebuah jaringan internet, Internet merupakan kepanjangan dari interconnected networking, yang mempunyai arti hubungan komputer dengan berbagai tipe yang membentuk sistem jaringan yang mencakup seluruh dunia (jaringan komputer global) dengan melalui jalur telekomunikasi seperti telepon, radio link, satelit dan lainnya. Istilah INTERNET berasal dari bahasa Latin inter, yang berarti “antara”. Internet adalah sebuah dunia maya jaringan komputer (interkoneksi) yang terbentuk dari milyaran komputer di dunia. Internet merupakan hubungan antar berbagai jenis komputer dan jaringan di dunia yang berbeda sistem operasi maupun aplikasinya di mana hubungan tersebut memanfaatkan kemajuan media komunikasi yang menggunakan protokol standar. protokol yang standar yang menyediakan akses internet memiliki banyak macamnya. Pada gambar 2.11 adalah teknologi untuk konektivitas IoT.



Gambar 2.11 Macam-macam konektivitas IoT

Sumber Gambar : [images \(286x176\) \(gstatic.com\)](https://images (286x176) (gstatic.com))

Pada *greenhouse* media komunikasi yang digunakan bersifat lokal dan membutuhkan kecepatan yang tinggi sehingga pada penelitian ini menggunakan jaringan wifi untuk menyediakan akses internet pada mikrokontroler.

Agar internet dapat berkomunikasi terdapat protokol internet. Protokol sendiri merupakan sebuah aturan atau standar yang mengatur atau mengijinkan terjadinya hubungan,

komunikasi, dan perpindahan data antara dua atau lebih titik komputer. Protokol dapat diterapkan pada perangkat keras, perangkat lunak atau kombinasi dari keduanya. Pada tingkatan yang terendah, protokol mendefinisikan koneksi perangkat keras [36]. Menurut penelitian yang dilakukan oleh Nindithia Putri Windryani, Dr. Nyoman Bogi A. K, S.T., MSEE., Ratna Mayasari, S.T., M.T yang berjudul COMPARISON ANALYSIS BETWEEN MQTT AND HTTP PROTOCOL IN PATRIOT IOT PLATFORM terdapat 2 protokol komunikasi yang terbaik yang dapat digunakan untuk IoT yakni MQTT dan HTTP berdasarkan hasil penelitian didapatkan bahwa protokol HTTP yang masih memiliki beberapa kekurangan yaitu pemakaian bandwidth yang cukup besar, ukuran paket yang besar sehingga tidak reliable untuk berjalan pada sistem yang memiliki bandwidth rendah atau latency yang tinggi. Berdasarkan kekurangan protokol HTTP tersebut, implementasi protokol MQTT server sangat dibutuhkan untuk menunjang pengembangan IoT Platform. MQTT merupakan protokol komunikasi yang sangat sederhana dan ringan. Protokol MQTT dapat lebih reliable berjalan pada keadaan bandwidth rendah atau latency tinggi dibandingkan protokol HTTP [37]. Sehingga protokol yang digunakan pada penelitian ini adalah MQTT.

2.2.9 Website

Website adalah sebuah kumpulan halaman yang berisi informasi tertentu dan dapat diakses oleh banyak orang melalui internet. Website dapat dibuka dengan menuliskan URL atau alamat website di browser. Keuntungan utama dalam mengadopsi web untuk pengembangan perangkat lunak diantaranya adalah minim biaya instalasi, peningkatan fitur baru untuk pengguna secara otomatis, dan akses secara universal dari semua perangkat yang terhubung dengan internet. Namun, terdapat sisi negatif dari web, yaitu rentan terjadinya error dan pengujian yang tidak mudah sehingga diperlukan pengujian yang memadai [38]. Berdasarkan keuntungan tersebut website dapat diakses secara universal sehingga sistem dapat dibuka juga di *smartphone* sehingga cocok untuk pengembangan awal sistem. Dalam membuat sebuah website hal yang perlu dipelajari yaitu :

1. HTML

HTML yang merupakan singkatan dari Hyper Text Markup Language adalah serangkaian kode program yang merupakan dasar dari representasi visual sebuah halaman Web. Didalamnya berisi kumpulan informasi yang disimpan dalam tag-tag tertentu, dimana tag-tag tersebut digunakan untuk melakukan format terhadap informasi yang dimaksud. Berbagai pengembangan telah dilakukan terhadap kode

HTML dan telah melahirkan teknologi-teknologi baru di dalam dunia pemrograman web. Kendati demikian, sampai sekarang HTML tetap berdiri kokoh sebagai dasar dari bahasa web seperti PHP, ASP, JSP dan lainnya. Bahkan secara umum, mayoritas situs web yang ada di Internet pun masih tetap menggunakan HTML sebagai teknologi utama mereka. [39]

2. Cascading Style Sheet

Cascading Style Sheet merupakan kepanjangan dari CSS. Penggunaan CSS membuat pemrograman Web menjadi lebih mudah karena kita dapat melakukan penyeragaman format terhadap elemen-elemen yang sama dalam situs dengan cepat. Saat ini hampir semua situs berbasis HTML menggunakan CSS untuk meningkatkan keluwesan tampilan. CSS dapat disimpan dalam file terpisah dengan ekstensi .css, dan setiap perubahan yang dilakukan pada file tersebut akan mempengaruhi seluruh dokumen HTML yang terkait padanya. Dengan demikian, waktu untuk melakukan perubahan terhadap situs dengan jumlah halaman yang banyak dapat dikurangi berkat bantuan CSS.

3. JavaScript

JavaScript adalah bahasa pemrograman web yang bersifat *Client Side Programming Language*. *Client Side Programming Language* adalah tipe bahasa pemrograman yang pemrosesannya dilakukan oleh client. Aplikasi client yang dimaksud merujuk kepada web browser seperti Google Chrome, Mozilla Firefox, Opera Mini dan sebagainya. JavaScript pertama kali dikembangkan pada pertengahan dekade 90'an. Meskipun memiliki nama yang hampir serupa, JavaScript berbeda dengan bahasa pemrograman Java. Untuk penulisannya, JavaScript dapat disisipkan di dalam dokumen HTML ataupun dijadikan dokumen tersendiri yang kemudian diasosiasikan dengan dokumen lain yang dituju. JavaScript mengimplementasikan fitur yang dirancang untuk mengendalikan bagaimana sebuah halaman web berinteraksi dengan penggunanya [40].

2.2.10 Front End

Front End Developer adalah pengembangan antarmuka pengguna grafis dari sebuah situs web, melalui penggunaan HTML, CSS, dan JavaScript, sehingga pengguna dapat melihat dan berinteraksi dengan situs web tersebut. *Front End Developer* ini adalah seorang yang bertugas mengelola apa yang dilihat pengguna di browser mereka.

Mereka merancang, menganalisis kode, dan men-debug sisi klien dari suatu aplikasi. Ini membuat mereka bertanggung jawab atas tampilan, nuansa, dan desain situs web atau aplikasi web. Seorang front end developer harus memastikan tampilan website sesuai dengan yang disepakati dengan desainer—baik dari segi tampilan maupun fungsionalitas.

Sederhana front end developer adalah pekerjaan dalam bentuk pemrograman yang mengelola dan mengembangkan tampilan sebuah aplikasi atau website. Adapun hal yang biasa dilakukan oleh front end developer yaitu mengkombinasikan teknik desain, teknologi, dan pemrograman untuk menghasilkan tampilan situs web yang menarik, interaktif, serta menangani debugging (masalah) yang terjadi. front end memiliki tanggung jawab untuk memastikan bahwa pengunjung dapat dengan mudah mengakses dan menggunakan sebuah aplikasi atau situs web. Contohnya, setiap kali kamu mengunjungi situs web, apa pun yang kamu lihat, klik, atau gunakan adalah pekerjaan front end. Mulai dari desain/tata letak, konten, tombol, gambar, navigasi, dan tautan internal. *Front end* akan bekerja sama dengan UI/UX terkait desain website yang diinginkan. Karena Tim UI/UX akan mendesain tampilan dari website dan kemudian mereka akan memberikan desain tersebut ke front end. Selanjutnya tim front end akan mengubah gambar desain dari UI/UX ke kode yang akan menghasilkan elemen visual pada tampilan website. Singkatnya, seorang *front end developer* mengaktifkan desain dan tombol-tombol pada sebuah web/aplikasi yang telah dirancang oleh UI/UX [41].

2.2.11 Backend

Back-end merupakan suatu program yang berjalan pada sisi server (server-side) yang melakukan tugas untuk berinteraksi langsung dengan basis data dalam melakukan manipulasi data ke basis data, sehingga *back-end* tidak melakukan interaksi secara langsung kepada pengguna. Dalam interaksi antar layanan client yang berbeda, salah satu teknologi yang digunakan adalah *Application Programming Interfaces* (API). API merupakan antarmuka yang digunakan untuk mengakses aplikasi atau layanan dari sebuah program [42].

API memungkinkan *developer* untuk memakai fungsi yang sudah ada dari aplikasi lain sehingga tidak perlu membuat ulang dari awal. Salah satu arsitektur *back-end* adalah *Representational State Transfer* (REST). REST merupakan seperangkat prinsip arsitektur yang melakukan transmisi data melalui antarmuka yang terstandarisasi seperti HTTP. REST API sendiri merupakan istilah yang dipakai untuk layanan web yang mengimplementasikan

arsitektur REST sebagai API. Untuk membangun sebuah back-end server, diperlukan penggunaan bahasa pemrograman yang berjalan pada sisi server (server-side) [43].

2.2.12 Database

Database atau basis data adalah kumpulan informasi yang disimpan di dalam komputer secara sistematis sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut. Perangkat lunak yang digunakan untuk mengelola dan memanggil kueri (query) basis data disebut sistem manajemen basis data (database management system, DBMS). Sistem basis data dipelajari dalam ilmu informasi. Konsep dasar dari basis data adalah kumpulan dari catatan-catatan, atau potongan dari pengetahuan. Sebuah basis data memiliki penjelasan terstruktur dari jenis fakta yang tersimpan di dalamnya: penjelasan ini disebut skema. Skema menggambarkan objek yang diwakili suatu basis data, dan hubungan di antara objek tersebut. Ada banyak cara untuk mengorganisasi skema, atau memodelkan struktur basis data: ini dikenal sebagai model basis data atau model data [44]. Untuk merancang basis data dapat membuat ER- Diagram atau enti

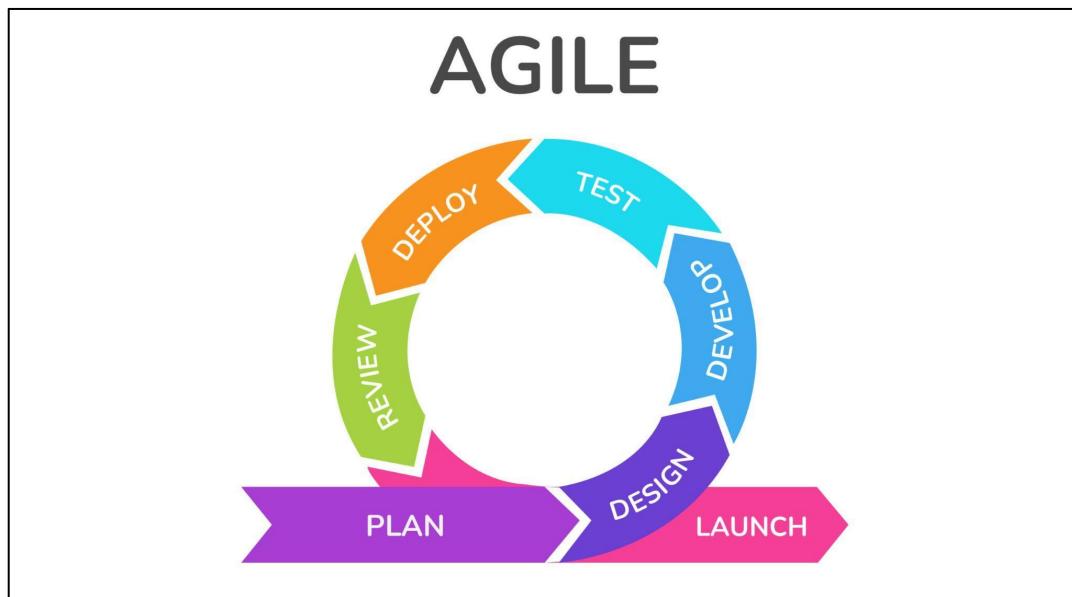
2.2.13 Framework

Framework adalah kerangka kerja. Framework juga dapat diartikan sebagai kumpulan script [45]. Framework adalah komponen pemrograman yang siap re-use (bisa digunakan ulang) kapan saja, sehingga programmer tidak harus membuat skrip yang sama untuk tugas yang sama. Misalkan programmer ingin halaman-halaman web menampilkan data dengan paginasi (paging) halaman, framework telah menyediakan fungsi paging tersebut sedangkan programmer cukup menggunakan fungsi tersebut pada saat coding, tetapi tentu dengan kaidah-kaidah yang ditetapkan oleh masing-masing framework [46].

2.2.14 Agile Kanban

Agile merupakan Konsep yang dicetuskan oleh Kent Beck dan 16 rekannya dengan menyatakan bahwa *agile software development* adalah cara membangun *software* dengan melakukannya dan membantu orang lain membangunnya sekaligus. *Agile software development methods* atau *agile methodology* merupakan sekumpulan metodologi pengembangan perangkat lunak yang berbasis pada pengembangan iteratif, di mana persyaratan dan solusi berkembang melalui kolaborasi antar tim yang terorganisir. Metode

agile merupakan metode pengembangan incremental yang fokus pada perkembangan yang cepat, perangkat lunak yang dirilis bertahap, mengurangi overhead proses, dan menghasilkan kode berkualitas tinggi dan pada proses perkembangannya melibatkan pelanggan secara langsung[47]. Gambaran dari proses agile pada gambar 2.12.



Gambar 2.12 Tahapan SDLC metode Agile

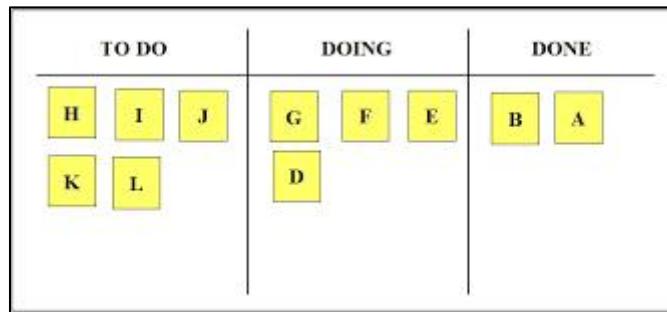
Sumber Gambar : https://global-uploads.webflow.com/6100d0111a4ed76bc1b9fd54/634e6a7f46239312f6d41ca2_agile%20method%205.jpeg

Kanban berasal dari bahasa Jepang untuk kartu visual atau kartu signal. Konsep kanban pertama kali digunakan dalam sistem manufaktur Toyota [48]. Cara kerja dari metode kanban berdasarkan penelitian yang telah berhasil dilakukan oleh anderson adalah agile kanban menggunakan sebuah *board* atau papan yang digunakan untuk melakukan visualisasi pada hal-hal yang akan dikerjakan agar pekerjaan saat mengembangkan sistem dapat terarah dimana sistem yang belum selesai ataupun sedang dikerjakan dan juga anderson juga telah mendefinisikan lima prinsip pada metode Agile Kanban [49], yaitu:

1. Membatasi pekerjaan yang sedang berlangsung
2. Memvisualisasikan alur kerja
3. Mengukur dan mengatur alur
4. Membuat kebijakan proses eksplisit
5. Menggunakan model untuk melihat peluang kemajuan.

Sehingga metode agile kanban membutuhkan *board* atau papan dibagi menjadi kolom-kolom atau tahapan, setiap tahapan merepresentasikan kondisi yang terjadi pada tugas yang ada, tugas direpresentasikan dengan sebuah kartu yang akan ditempel ke sebuah papan

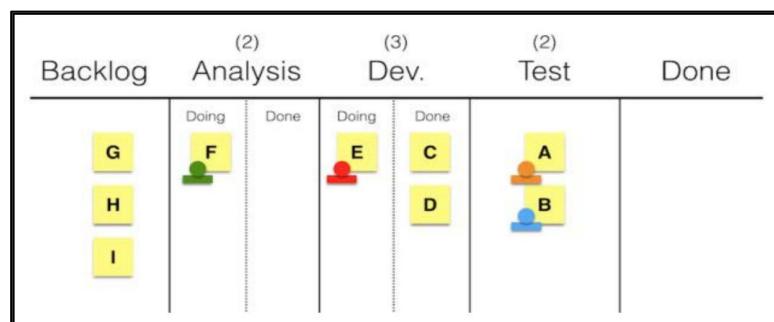
sehingga setiap tugas dapat diketahui sedang berada di tahapan yang ada di papan. Di dalam papan kanban kartu ataupun tugas yang ada akan berpindah dari kiri ke kanan, jika sudah paling kanan menyatakan tugas yang ada sudah selesai. Menurut penelitian yang dilakukan oleh Nakazawa dan Tanaka terdapat 2 macam tipe kanban yakni pada *simple kanban board* seperti yang merepresentasikan 3 tahapan tugas yang ada, sedang dikerjakan dan sudah dikerjakan seperti pada gambar 2.13.



Gambar 2.13 Kanban Simple board

Sumber gambar : The State of the Art of Agile Kanban Method: Challenges and Opportunities[50].

Tipe kedua adalah *detailed kanban board* memiliki jumlah tahapan yang lebih banyak yakni seperti gambar 2.14 [50] :



Gambar 2.14 Detailed Kanban Board

Sumber gambar : The State of the Art of Agile Kanban Method: Challenges and Opportunities[50].

Sehingga dapat disimpulkan untuk metode kanban alur atau tahapan pengembangan yang akan digunakan dalam penelitian ini adalah yakni sebagai berikut :

1. Backlog/To Do

Backlog adalah daftar hal-hal yang menurut kami harus kami lakukan, karena berbagai alasan (klein, kebutuhan kami sendiri, dll.). Apalagi diprioritaskan, jadi hal-

hal di atas adalah hal-hal yang benar-benar ingin (atau perlu) kita lakukan sedangkan hal-hal dibawah mungkin adalah fantasi yang tidak akan pernah terjadi.

2. Development/Doing

Pada tahapan doing merupakan tahap pengembangan dimana tahapan ini seperti pembuatan ui/ux ataupun coding.

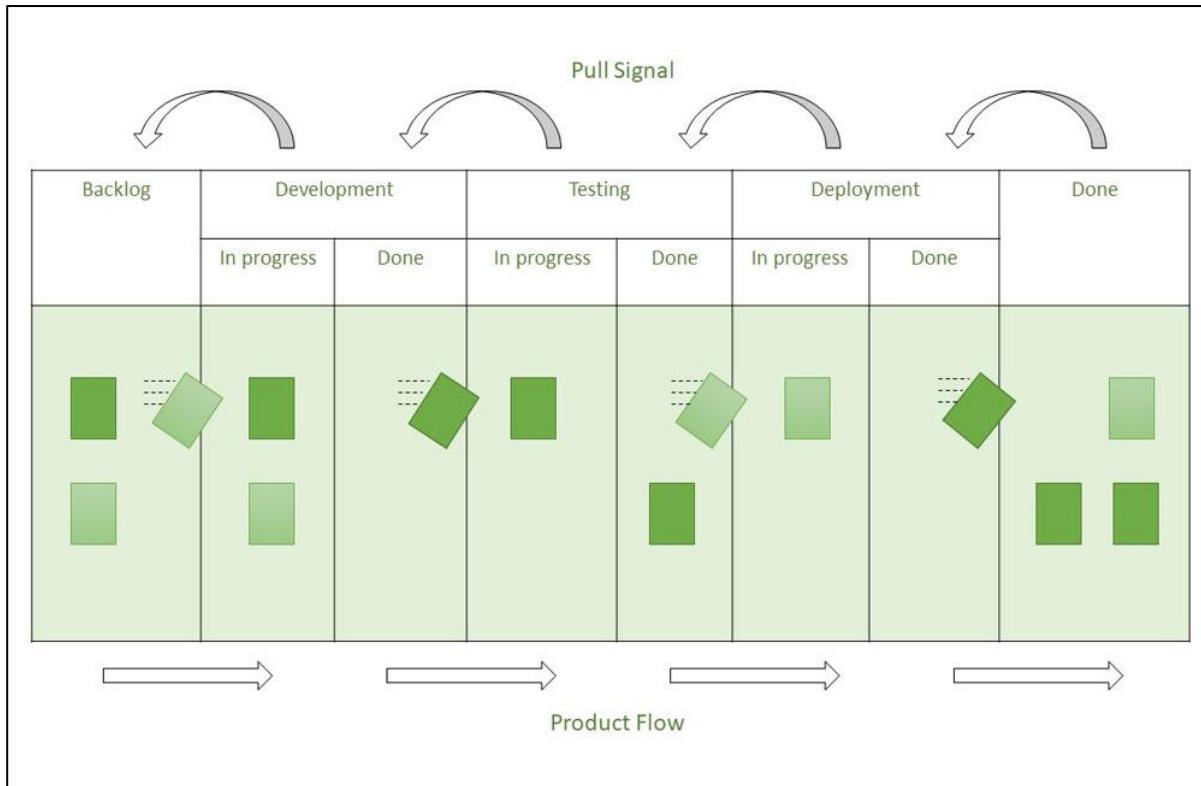
3. Testing

Testing merupakan tahapan menguji tugas yang telah dikerjakan sehingga program saat digunakan bekerja sesuai dengan yang diinginkan dan terhindar dari bug. Terdapat banyak metode dalam melakukan pengujian terhadap kode yang kita buat salah satunya adalah blackbox testing dan juga whitebox testing. Kedua testing ini memiliki kelebihan dan kekurangannya masing masing. Pada penelitian ini akan menggunakan black-box testing menguji hasil dari pengembangan jika terjadi kesalahan maka akan kembali ke tahap development untuk memperbaiki tugas ataupun kartu yang telah dikerjakan

4. Done

Done merupakan tahapan dimana kartu ataupun tugas sudah selesai dikerjakan.

4 hal tersebut akan terus berjalan hingga seluruh backlog selesai jika backlog selesai maka akan masuk pada tahap produksi yakni *deploy* dari program. Pada gambar 2.15 merupakan alur dari pembuatan agile kanban



Gambar 2.15 Alur Agile Kanban
 Sumber gambar: <https://media.geeksforgeeks.org/wp-content/uploads/20220310192308/knabanpullbasedsystem.jpg>

2.2.15 Black Box Testing

Perangkat lunak yang dirancang harus melalui tahap pengujian untuk memastikan kualitas dari perangkat lunak. Pengujian dilakukan untuk menemukan kesalahan yang ada di suatu perangkat lunak. Pengujian kotak hitam (black box testing) merupakan pengujian yang dilakukan dengan mengamati hasil eksekusi melalui data uji dan memeriksa fungsional dari perangkat lunak tanpa mengetahui kode program yang digunakan. Ada beberapa metode pada black box testing yang dapat dilakukan seperti *Cause Effect Relationship Testing*, *Equivalence Partitions* dan *Boundary Value Analysis* [51]. Terdapat alasan dalam penggunaan black box dalam penelitian ini yakni Pengembangan yang lebih cepat dan memakan cost yang lebih sedikit dikarenakan dalam white box dibutuhkan seseorang yang dapat memahami code dan pembuatan unit test. Selain itu juga dapat mengetahui kesalahan yang tidak terdeteksi secara kode tetapi secara fungsionalitas tidak berjalan sesuai yang diinginkan [52]. Sehingga, penelitian ini akan menggunakan metode black-box testing dikarenakan selain perangkat lunak terdapat perangkat keras yang perlu bekerja sesuai fungsionalitas yakni membaca dari sensor sehingga metode ini cocok untuk pembuatan IoT.

2.2.16 Pengujian

System Usability Scale (SUS) merupakan kuesioner yang dapat digunakan untuk mengukur usability sistem komputer menurut sudut pandang subyektif pengguna. SUS dikembangkan oleh John Brooke. Hingga saat ini, SUS banyak digunakan untuk mengukur usability dan menunjukkan beberapa keunggulan, antara lain SUS dapat digunakan dengan mudah, karena hasilnya berupa skor 0–100, SUS sangat mudah digunakan, tidak membutuhkan perhitungan yang rumit, SUS tersedia secara gratis, tidak membutuhkan biaya tambahan, dan SUS terbukti valid dan reliable, walau dengan ukuran sampel yang [53]. Cara untuk menggunakan *System Usability Scale* (SUS) adalah dengan membuat kuesioner yang dapat digunakan untuk mengukur usability perangkat lunak yang telah dibuat menurut sudut pandang subyektif pengguna yang terdiri dari 10 item pertanyaan. SUS terdiri dari 10 pertanyaan sebagai berikut [54]:

Tabel 2.7 Pengujian SUS

| No | Pertanyaan |
|----|--|
| 1 | Saya berpikir akan menggunakan sistem ini lagi |
| 2 | Saya merasa sistem ini rumit untuk digunakan |
| 3 | Saya merasa sistem ini mudah digunakan |
| 4 | Saya membutuhkan bantuan dari orang lain atau teknisi dalam menggunakan sistem ini |
| 5 | Saya merasa fitur-fitur sistem ini berjalan dengan semestinya |
| 6 | Saya merasa ada banyak hal yang tidak konsisten (tidak serasi pada sistem ini) |
| 7 | Saya merasa orang lain akan memahami cara menggunakan sistem ini dengan cepat |
| 8 | Saya merasa sistem ini membingungkan |
| 9 | Saya merasa tidak ada hambatan dalam menggunakan sistem ini |
| 10 | Saya perlu membiasakan diri terlebih dahulu sebelum menggunakan sistem ini |

Berdasarkan 10 pertanyaan setiap pertanyaan perlu diisikan dengan skor dari 1-5 Perhitungan dari 10 pertanyaan adalah :

- Untuk setiap pertanyaan bernomor ganjil, hasil skornya dikurangi angka 1.

[Penilaian pengguna – 1 = skor pertanyaan]

2. Untuk setiap pertanyaan bernomor genap, maka kita harus mengurangi angka 5 dengan hasil skornya.

[5 - Penilaian pengguna = skor pertanyaan]

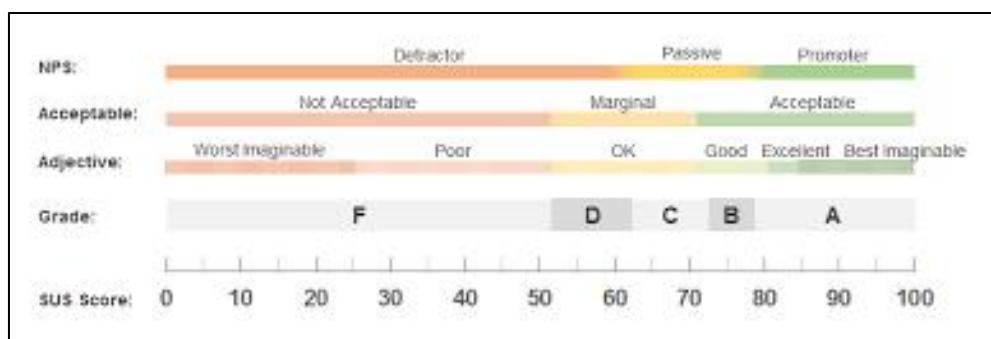
3. Kemudian jumlahkan semua hasil skor dari setiap pertanyaan per responden, kemudian hasilnya dikalikan dengan angka 2,5.

[[Skor pertanyaan ke 1] + [Skor pertanyaan ke 2] + ... + [Skor pertanyaan ke n] * 2.5
= skor responden]

4. Jumlahkan semua hasil skor setiap responden yang telah melalui langkah 1 hingga 3 diatas, kemudian hitung nilai rata-ratanya.

[Total skor responden] / jumlah responden = Hasil Skor SUS

Hasil skor SUS akan dapat dilihat skala sistem yang telah dibuat apakah sudah baik ataupun tidak dapat diterima user. Berikut adalah skala interpretasi dari skor SUS pada gambar 2.15 [55]:



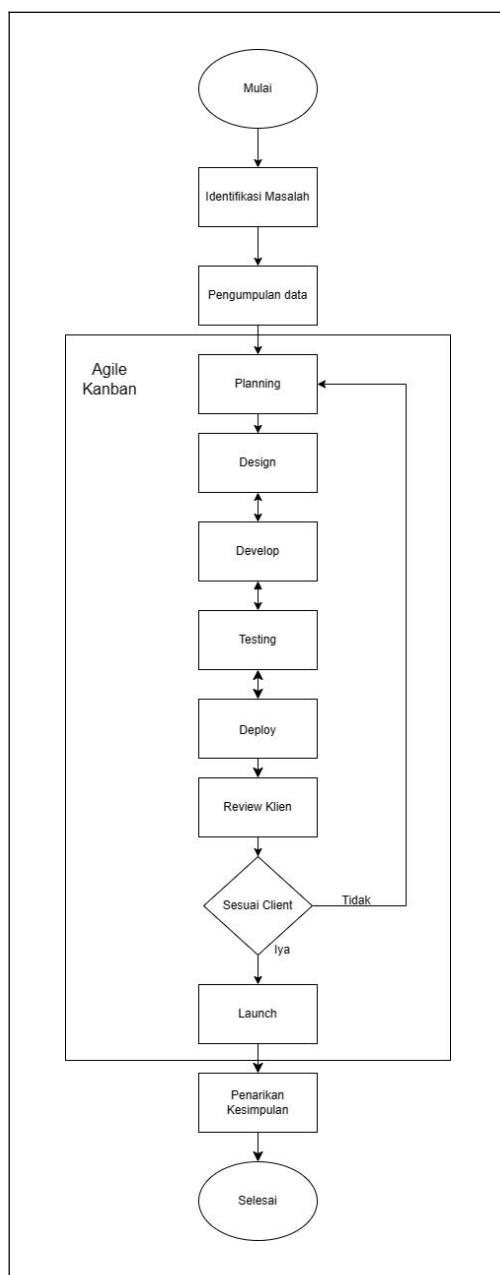
Gambar 2.16 Skala Interpretasi SUS

BAB III

METODOLOGI PENELITIAN

3.1 Alur Penelitian

Dalam penelitian ini, dilakukan dengan menggunakan alur penelitian untuk memudahkan peneliti dalam rancang bangun sistem. Alur penelitian dibangun dengan diagram alir yang dapat dilihat pada Gambar 3.1.



Gambar 3.1 Alur Penelitian

3.2 Penjabaran Langkah Penelitian

Dari alur penelitian yang sudah digambarkan, berikut merupakan penjelasan setiap langkah dalam alur penelitian tersebut.

3.2.1 Identifikasi Masalah

Pada tahapan ini akan dilakukan analisis permasalahan yang terjadi pada pertumbuhan melon yang kurang baik.. Masalah yang teridentifikasi yaitu kondisi lingkungan tanaman yang tidak sesuai dengan kondisi optimal dan pemberian nutrisi yang tidak tepat. Oleh karena itu, untuk menyelesaikan permasalahan yang ada dilakukan pengumpulan data pada sub bab 3.2.2 untuk mendukung solusi dalam pemecahan masalah pada penelitian in

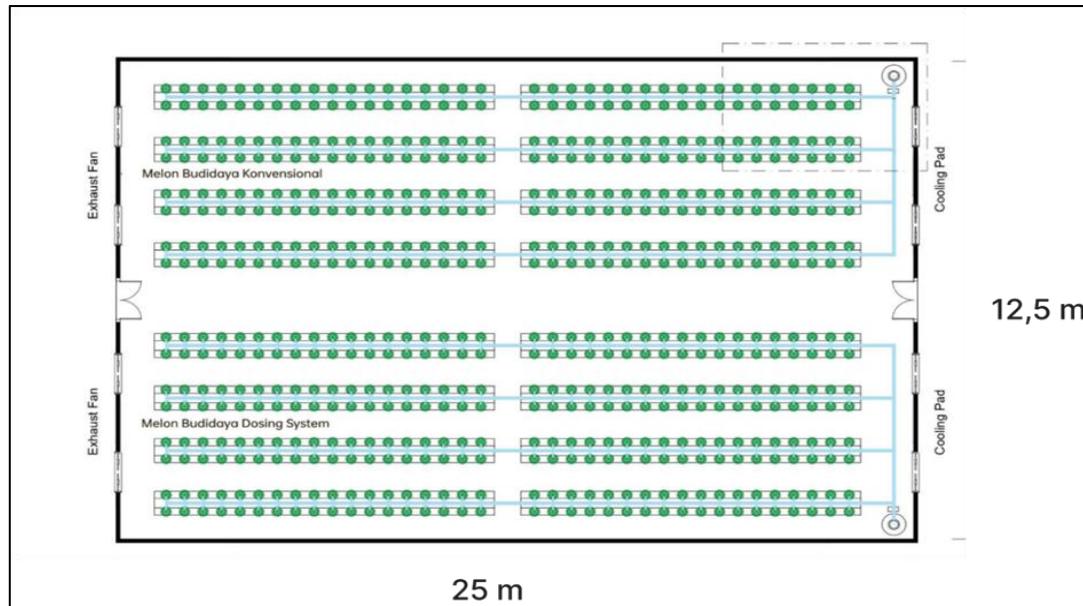
3.2.2 Pengumpulan Data

Perancangan pembuatan sistem memerlukan pemahaman secara teoritis terhadap pembuatan IoT serta website secara berkala dari berbagai referensi jurnal dan buku. Hal ini dilakukan agar memiliki gambaran dalam memilih komponen serta alat yang dibutuhkan. Sehingga diharapkan dalam pelaksanaannya tidak membutuhkan banyak pengeluaran serta dapat mengatasi berbagai masalah dalam pengembangan secara tim.

1. Data Primer

Data primer merupakan data yang diperoleh langsung dari sumber pertama atau objek penelitian, data primer pada penelitian ini dikumpulkan menggunakan metode wawancara dan juga Observasi pada *Greenhouse* ITERA. Wawancara dilakukan kepada pemilik dari tanaman melon dan juga *Greenhouse* yaitu ibu Zunanik Mufidah, S.TP., M.Si.

Data primer bertujuan untuk mendapatkan informasi mengenai kebutuhan sistem yang akan dibangun dari pengguna yang akan menggunakan sistem. Pada wawancara yang dilakukan terdapat beberapa hal yang diperlukan untuk melakukan monitoring *greenhouse* yakni *greenhouse* memiliki ukuran 25m x 12.5m. *Greenhouse* memiliki sistem pengendali lingkungan yakni kipas dan juga *cooling pad* sehingga diperlukan sistem otomatis agar kipas dan *cooling pad* dapat menyala berdasarkan sensor suhu. seperti pada gambar 3.2.



Gambar 3.2 Ilustrasi *Greenhouse* ITERA

Dilakukan observasi pada *greenhouse* dengan menggunakan alat ukur didapatkan Suhu sehingga didapatkan bahwa suhu yang terjadi pada *greenhouse* mengalami kenaikan setiap 3 menit sekali seperti pada tabel 3.1 berikut akan memiliki perbedaan 0.1 derajat.

Tabel 3.1 Perubahan Suhu pada *Greenhouse*

| Parameter: Suhu | |
|-----------------|---------|
| Waktu | Nilai |
| 11.00.10 | 28.85 C |
| 11.00.20 | 28.84 C |
| 11.00.30 | 28.83 C |
| 11.00.40 | 28.82 C |
| 11.00.50 | 28.81 C |
| 11.01.00 | 28.81 C |
| 11.02.00 | 28.81 C |
| 11.03.15 | 28.79 C |
| 11:05:00 | 28.7 C |

Dilakukan observasi pada *greenhouse* didapatkan Kelembaban bahwa pada *greenhouse* mengalami kenaikan yang signifikan pada setiap 4 menit sekali seperti pada tabel 3.2 berikut terjadi perubahan signifikan mulai dari 68.67 menjadi 70.15 dengan jeda waktu 4 menit.

Tabel 3.2 Perubahan Kelembaban pada *Greenhouse*

| Parameter: Kelembaban | |
|-----------------------|--------|
| Waktu | Nilai |
| 07:28 | 70.76% |
| 07:29 | 69.67% |
| 07:30 | 69.53% |
| 07:31 | 69.83% |
| 07:32 | 69.61% |
| 07:33 | 70.15% |
| 07:34 | 70.3% |

Dilakukan observasi pada *greenhouse* didapatkan perubahan debit bahwa pada *greenhouse* mengalami kenaikan pada setiap 2 detik sekali seperti pada tabel 3.3.

Tabel 3.3 Perubahan Debit Air pada *Greenhouse*

| Parameter: debit air | |
|----------------------|--------|
| Waktu | Nilai |
| 07:39:31 | 0 ml |
| 07:39:33 | 30 ml |
| 07:39:35 | 60 ml |
| 07:39:39 | 190 ml |
| 07:39:41 | 380 ml |
| 07:39:43 | 570 ml |
| 07:39:45 | 760 ml |
| 07:39:47 | 940 ml |

2. Data Sekunder

Data sekunder diperoleh dari jurnal, artikel atau sumber lain yang dapat mendukung penelitian ini. Kegiatan studi pustaka ini bertujuan untuk memperoleh informasi dari penelitian terdahulu yang digunakan sebagai bahan pertimbangan dalam melakukan penelitian *monitoring greenhouse*. Terdapat data-data yang dibutuhkan untuk menyalakan atau mematikan sensor menurut penelitian yang dilakukan oleh Hidayat et al terdapat parameter yang memiliki Batasan pada saat penanaman melon yang dapat dilihat pada tabel 3.4 [56].

Tabel 3.4 Hubungan antara Sensor dan Kendali

| Parameter | Kendali | Batasan |
|---|----------------|-----------------------|
| Suhu | Exhaust Fan | 25C - 30 C |
| Kelembaban | Cooling Pad | 70% - 80% |
| Nutrisi Vegetative (14-50 hari) | Selenoid Valve | 600 ml/tanaman/ hari |
| Nutrisi Generatif (50 hari setelah tanam) | Selenoid Valve | 1500 ml/tanaman/ hari |

3.2.3 Implementasi Sistem dengan Metode Agile Kanban

Berdasarkan permasalahan yang telah diperoleh akan dibuat sebuah IOT yang diharapkan dapat memberikan solusi bagi pengguna mulai dari Sensor, Sistem Kendali dan Juga website mulai dari desain ui ux, front end hingga backend. Berdasarkan Data-data yang telah dikumpulkan Sistem IOT akan dibuat dengan menggunakan metode SDLC *Agile Kanban* dengan tahapan planning membuat backlog seperti user story dan kebutuhan dari sistem, setelah itu masuk ke tahap desain mendesain seluruh kebutuhan untuk masuk ke tahap selanjutnya implementasi dan terakhir testing.

3.2.4 Penarikan Kesimpulan

Berdasarkan sistem yang telah dibuat berdasarkan tujuan dari penelitian diperlukan penarikan kesimpulan dari penelitian yang telah dilakukan sehingga dapat mengetahui hasil dari penelitian yang telah dilakukan.

3.3 Alat dan Bahan

a. Alat

Adapun alat yang digunakan untuk membuat perangkat lunak yaitu :

1. Perangkat keras untuk Pengembang

1. Laptop

Spesifikasi laptop yang digunakan untuk mengembangkan perangkat lunak dapat dilihat pada Tabel 3.5.

Tabel 3.5 Spesifikasi Laptop

| Kategori | Tipe |
|----------------|------------------------|
| Sistem Operasi | Windows 11 Home 64-bit |
| CPU | AMD Ryzen 5 3500u |
| Memory | 8 GB |
| VGA | AMD Radeon vega 8 |
| Disk | 512 GB ssd nvme |

2. Perangkat lunak untuk pengembang

Adapun daftar dari perangkat lunak yang akan digunakan pada penelitian ini. Rincian dari perangkat lunak yang digunakan dapat dilihat pada Tabel 3.6.

Tabel 3.6 Perangkat Lunak yang digunakan

| No | Perangkat Lunak |
|----|--------------------------|
| 1 | Visual Studio code |
| 2 | Browser (Microsoft Edge) |
| 3 | Dbeaver |
| 5 | Postman |
| 6 | Draw.io |
| 7 | Git/github |

b. Bahan

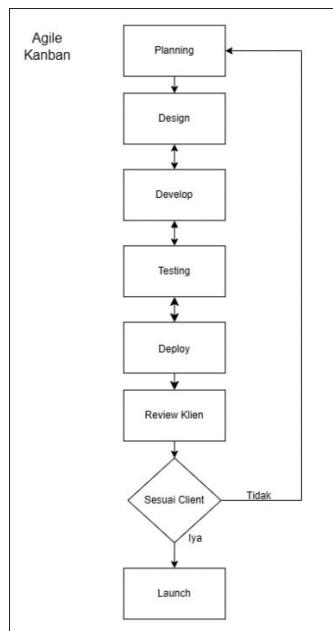
Bahan berisi mengenai bahan yang akan digunakan untuk penelitian, bahan yang digunakan untuk melakukan Adapaun bahan yang digunakan dalam penelitian ini dapat dilihat pada tabel 3.7.

Tabel 3.7 Bahan Penelitian

| No | Nama Alat | jumlah | harga satuan | harga |
|-------|-----------------------|---------|--------------|------------|
| 1 | sensor sht 10 | 1 | Rp.39.000 | Rp.39.000 |
| 2 | esp 8266 | 5 | Rp.32.000 | Rp.128.000 |
| 3 | Relay 5v | 2 | Rp.13,000 | Rp13,000 |
| 4 | Sensor Debit | 1 | Rp90,000 | Rp90,000 |
| 5 | Selenoid valve 1 inch | 1 | Rp 176,000 | Rp 176,000 |
| 6 | adapter robot 3A | 4 | Rp75,000 | Rp75,000 |
| 7 | PCB | 5 | Rp5.000 | Rp. 25.000 |
| 8 | Kabel pita | 2 meter | Rp. 6000 | Rp. 12.000 |
| Total | | | | Rp.558.000 |

3.4 Metode Pengembangan

Penelitian ini secara umum merupakan penelitian yang bertujuan untuk mengembangkan suatu produk yakni Pembuatan IOT sistem *monitoring greenhouse* menggunakan metode *Agile Kanban*. Metode Agile Kanban merupakan metode pengembangan perangkat lunak yang digunakan merupakan metode yang flexible dan cepat sehingga dalam pengembangan yang memiliki sistem cukup kompleks dan memiliki kemungkinan untuk adanya tambahan fitur sehingga diperlukan metode yang dapat digunakan secara fleksibel dan cepat. Dalam menggunakan metode *agile kanban* terdapat tahapan yang akan dilalui dalam metode *Agile Kanban* dapat dilihat pada gambar 3.3.



Gambar 3.3 Alur Agile Kanban

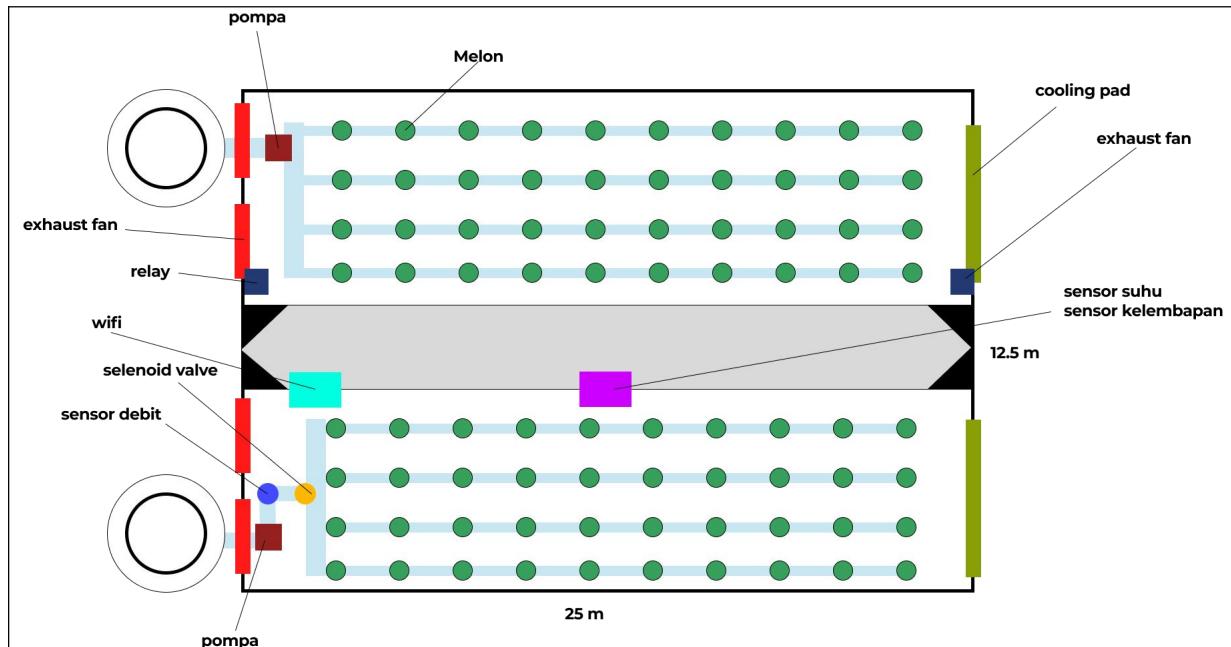
Penjelasan mengenai tahapan pada metode agile kanban lebih rinci dijelaskan pada sub bab 3.4.1.

3.4.1 Planning

Pada tahapan ini akan dilakukan pengidentifikasi permasalahan dan menganalisis kebutuhan sistem monitoring *greenhouse* yang akan dibangun pada *greenhouse* ITERA. Pada tahapan perencanaan ini akan dilakukan pembuatan user stories dan rancangan perangkat keras yang dibuat sehingga dapat menjadi sebagai acuan dalam pembuatan fitur pada sistem. Tahapan yang akan dilakukan pada perencanaan sebagai berikut.

3.4.1.1 Rancangan Perangkat Keras

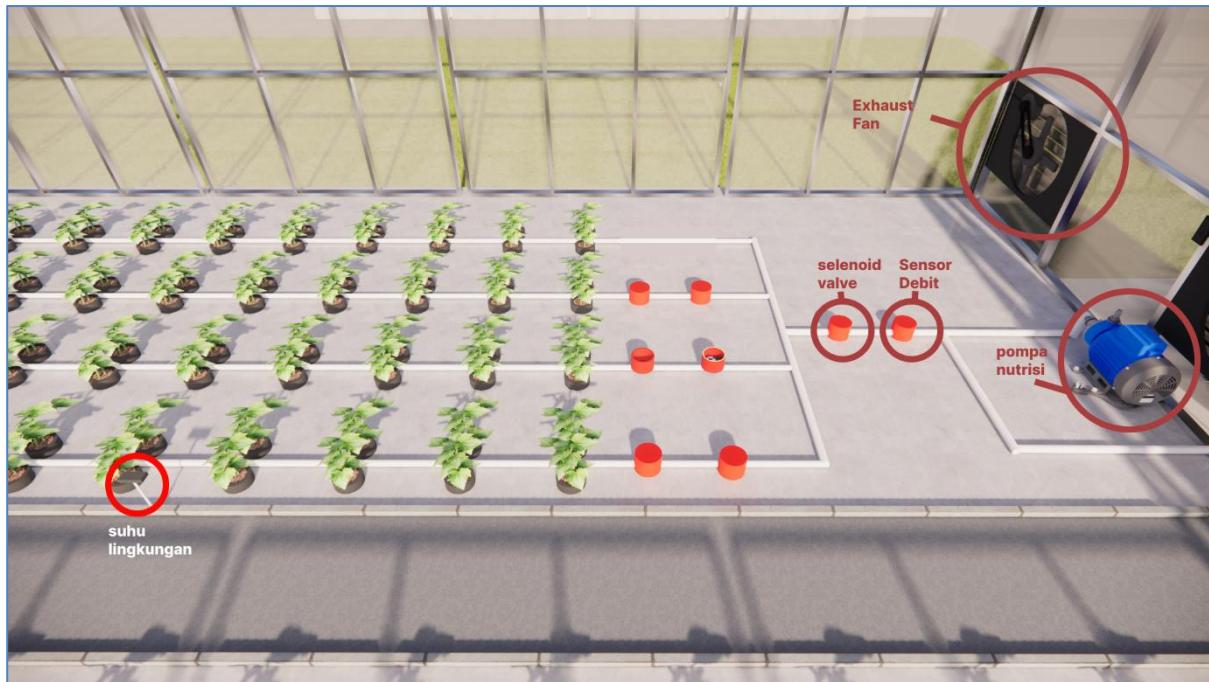
Pada tahapan perancangan perangkat keras berisi mengenai sensor-sensor yang akan digunakan dan dipakai untuk *greenhouse* berdasarkan data primer untuk mengendalikan suhu kelembaban dan jumlah nutrisi dengan *greenhouse*. Berikut adalah gambaran dari sensor dan aktuator yang akan digunakan. Dikarenakan akan dilakukan perbandingan menggunakan alat otomatis dan manual maka akan dibedakan menjadi yang bawah pada gambar atau kanan dari *greenhouse* menggunakan sensor dan yang bagian atas pada gambar atau kiri dari *greenhouse* akan menggunakan cara manual. Gambar detail dari perancangan *greenhouse* dapat dilihat pada gambar 3.4.



Gambar 3.4 Rancangan Pembuatan IoT pada *Greenhouse* ITERA

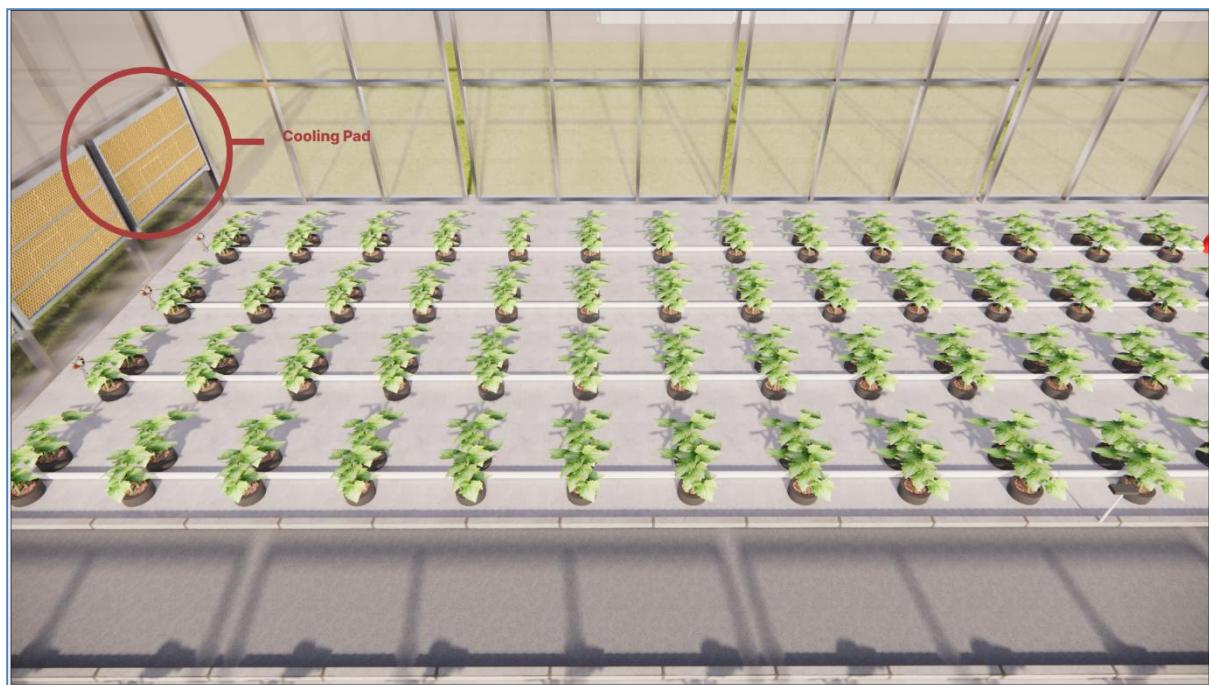
Sensor suhu terletak ditengah dikarenakan terletak jauh dari kipas dan juga jauh dari cooling pad sehingga suhu *greenhouse* yang terjadi lebih merata dan titik tengah menjadi titik yang jauh dari coling pad dan juga kipas. Serta untuk melakukan kendali pada nutrisi menggunakan seleoid valve dan juga sensor debit dan terakhir untuk jaringan yang digunakan wifi yang terletak dekat dengan kipas dikarenakan jika dekat dengan cooling pad lebih lembab dan dapat terkena air dari cooling pad. Untuk jaringan yang digunakan pada sistem monitoring *greenhouse* adalah dengan menggunakan *orbit* yakni wifi menggunakan kartu sim.

Terdapat tampilan dari samping *greenhouse* memiliki 4 *exhaust* dan juga terdapat pompa nutirisi yang mengalir dan juga untuk debit dan solenoid yang digunakan yang pertama dan untuk sensor lingkungan terletak di tengah *greenhouse* tampilan dari rancangan *greenhouse* dapat dilihat pada gambar 3.5.



Gambar 3.5 Rangan Greenhouse pada bagian amping kanan

Untuk sisi bagian kanan terdapat *cooling pad* yang berfungsi untuk menjaga kelembaban dan juga suhu dari greenhouse berbentuk kotak-kotak dan kardus tampilan dari rancangan *greenhouse* dapat dilihat pada gambar 3.6.



Gambar 3.6 Rangan Greenhouse pada bagian amping kanan

3.4.1.2 Rancangan Perangkat Lunak

3.4.1.2.1 User Story

Tabel *user story* digunakan untuk menjelaskan mengenai hal yang dibutuhkan oleh pengguna di dalam sistem yang sedang dikembangkan. *User story* didapatkan berdasarkan kebutuhan dari pengguna ataupun dari klien pada sistem. Berikut merupakan tabel *user story* untuk pengembangan proyek ini pada tabel 3.8.

Tabel 3.8 User Story

| Kode | Sebagai | Ingin | Sehingga |
|-------|---------------------------|---|--|
| US-01 | Penjaga <i>Greenhouse</i> | Melakukan login ke dalam website | Dapat melakukan hak akses sebagai admin yang mampu mengelola sensor dan aktuator dalam <i>greenhouse</i> |
| US-02 | Penjaga <i>Greenhouse</i> | Menambahkan, menghapus, mengubah data <i>greenhouse</i> | Dapat mengelola data <i>greenhouse</i> pada sistem |
| US-03 | Penjaga <i>Greenhouse</i> | Menambahkan, menghapus, mengubah data sensor | Dapat mengelola data sensor yang ada |
| US-04 | Penjaga <i>Greenhouse</i> | Menambahkan, menghapus, mengubah data aktuator | Dapat mengelola data aktuator yang ada |
| US-05 | Penjaga <i>Greenhouse</i> | Mengetahui notifikasi yang masuk dari kamera pendekripsi pada <i>greenhouse</i> | Dapat melihat notifikasi dari sistem |
| US-06 | Penjaga <i>Greenhouse</i> | Mengetahui data yang | Dapat melihat nilai |

| Kode | Sebagai | Ingin | Sehingga |
|-------|---------------------------|---|---|
| | | ditangkap sensor dalam <i>greenhouse</i> | yang ditangkap oleh sensor dalam <i>greenhouse</i> |
| US-07 | Penjaga <i>Greenhouse</i> | Mengetahui data-data lampau yang ditangkap sensor | Dapat memperoleh informasi data-data lampau yang telah dirata-ratakan |
| US-08 | Penjaga <i>Greenhouse</i> | Mengetahui status <i>online/offline</i> dari sensor | Dapat melihat status <i>online/offline</i> dari sensor |
| US-09 | Penjaga <i>Greenhouse</i> | Mengetahui status <i>online/offline</i> dari aktuator | Dapat melihat status <i>online/offline</i> dari aktuator |
| US-10 | Penjaga <i>Greenhouse</i> | Mematikan dan menyalakan aktuator yang ada di dalam <i>greenhouse</i> | Dapat mematikan dan menyalakan aktuator yang ada di dalam <i>greenhouse</i> |
| US-11 | Penjaga <i>Greenhouse</i> | Menghapus notifikasi | Dapat menghapus notifikasi |
| US-12 | Penjaga <i>Greenhouse</i> | Mengalihkan kendali menjadi otomatis | Dapat merubah sistem kendali dari manual menjadi otomatis |
| US-13 | Penjaga <i>Greenhouse</i> | Menambahkan kendali otomatis berdasarkan sensor | Dapat merubah sistem Kendali otomatis berdasarkan sensor |
| US-14 | Penjaga <i>Greenhouse</i> | Menambahkan kendali otomatis berdasarkan penjadwalan | Dapat menambahkan sistem Aktuator menjadi otomatis berdasarkan jadwal |
| US-15 | Penjaga <i>Greenhouse</i> | Menghapus kendali otomatis berdasarkan sensor | Dapat menghapus sistem Kendali otomatis berdasarkan sensor |
| US-16 | Penjaga <i>Greenhouse</i> | Menghapus kendali | Dapat menghapus |

| Kode | Sebagai | Ingin | Sehingga |
|-------|---------------------------|--|--|
| | | otomatis berdasarkan penjadwalan | sistem Kendali otomatis berdasarkan penjadwalan |
| US-17 | Penjaga <i>Greenhouse</i> | Menampilkan data historis Aktuator menyala | Dapat menampilkan data historis Aktuator menyala |
| US-18 | Penjaga <i>Greenhouse</i> | Menampilkan monitoring suhu | Dapat menampilkan suhu pada <i>greenhouse</i> |
| US-19 | Penjaga <i>Greenhouse</i> | Menampilkan monitoring Kelembaban | Dapat menampilkan suhu pada <i>greenhouse</i> |
| US-20 | Penjaga <i>Greenhouse</i> | Menampilkan monitoring Kelembaban | Dapat menampilkan suhu pada <i>greenhouse</i> |
| US-21 | Penjaga <i>Greenhouse</i> | Menampilkan monitoring debit | Dapat menampilkan suhu pada <i>greenhouse</i> |
| US-22 | Penjaga <i>Greenhouse</i> | Melakukan kontrol solenoid valve | Dapat mengontrol solenoid valve |
| US-23 | Penjaga <i>Greenhouse</i> | Melakukan kontrol exhaust fan | Dapat mengontrol exhaust fan |
| US-24 | Penjaga <i>Greenhouse</i> | Melakukan kontrol cooling pad | Dapat mengontrol cooling pad |

3.4.1.2.2 Kebutuhan Fungsional

Kebutuhan fungsional adalah jenis kebutuhan yang berisi mengenai fitur-fitur yang dapat dilakukan oleh pnegguna dalam sistem atau biasa disebut dengan fitur yang perlu di dalam perangkat lunak hal ini bertujuan pada saat pengembang seorang *developer* dapat mengerjakan sistem dengan sesuai. berikut adalah kebutuhan fungsional dari sistem monitoring *greenhouse* pada tabel 3.9.

Tabel 3.9 Kebutuhan Fungsional

| Kode | Kode User Story | Deskripsi |
|-------|-----------------|------------------------------|
| KF-01 | US-01 | Sistem mampu melakukan login |

| Kode | Kode User Story | Deskripsi |
|-------|--------------------------------|--|
| KF-02 | US-02 | Sistem mampu mengelola <i>greenhouse</i> |
| KF-03 | US-03 | Sistem mampu mengelola sensor |
| KF-04 | US-04 | Sistem mampu mengelola aktuator |
| KF-05 | US-05 | Sistem mampu menampilkan riwayat notifikasi |
| KF-06 | US-06, US-18,US-19,US-20,US-21 | Sistem mampu menampilkan data sensor |
| KF-07 | US-07 | Sistem mampu menampilkan grafik riwayat sensor |
| KF-08 | US-08 | Sistem mampu menampilkan sensor offline dan online |
| KF-09 | US-10,US-22,US-23,-24 | Sistem dapat mematikan dan menyalakan aktuator |
| KF-10 | US-9 | Sistem mampu menampilkan aktuator offline dan online |
| KF-11 | US-11 | Sistem mampu menghapus notifikasi |
| KF-12 | US-12 | Sistem mampu mengubah kontrol manual jadi otomatis atau sebaliknya |
| KF-13 | US-13, US-15 | Sistem mampu mengelola Aktuator secara otomatis berdasarkan sensor |
| KF-14 | US-14, US-16 | Sistem mampu mengelola Aktuator secara otomatis dengan penjadwalan |
| KF-15 | US-17 | Sistem mampu menampilkan riwayat aktuator hidup ataupun mati |

3.4.1.2.3 Kebutuhan Non Fungsional

Kebutuhan Non Fungsional dibuat untuk menentukan kebutuhan yang berisi mengenai keterbatasan dari layanan sistem atau dapat dikatakan juga sebagai fungsi yang ditawarkan sistem. kebutuhan Non Fungsional pada perangkat lunak sistem dapat dilihat pada table 3.11.

Tabel 3.10Non Fungsional

| Id | Parameter | Non-Fungsionalitas |
|---------|--------------|---|
| IHNF-01 | Portability | <i>Website</i> dapat dibuka jika pengguna terhubung dengan internet |
| IHNF-02 | Availability | <i>Website</i> dapat diakses selama 24 jam dalam sehari |

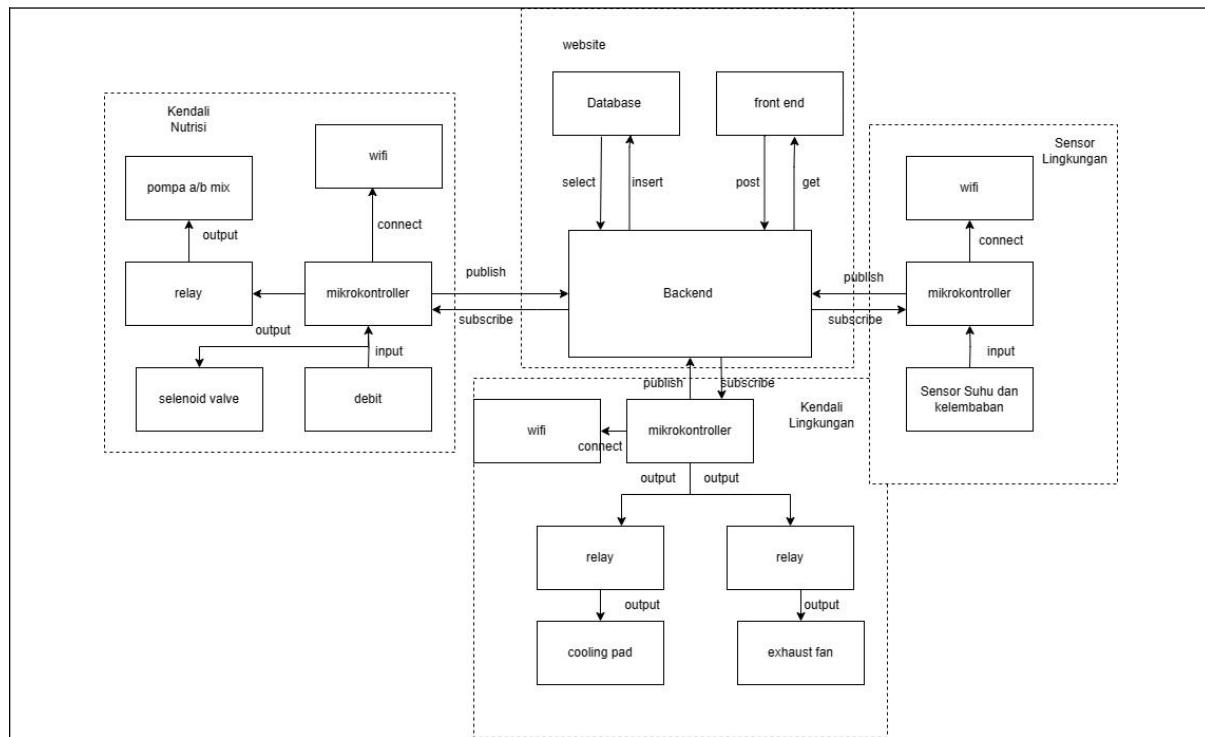
| | | |
|---------|----------|--------------------------------------|
| IHNF-03 | Security | <i>Website menggunakan token jwt</i> |
|---------|----------|--------------------------------------|

3.4.2 Design

Pada tahapan desain berfokus pada perencanaan *sistem* dengan pembuatan diagram yang akan membantu dalam pembuatan website, seperti diagram uml untuk pembuatan sistem

3.4.2.1 Diagram Blok

Diagram Blok merupakan diagram yang menggambarkan sistem secara utuh mulai dari penggunaan perangkat keras dan juga perangkat lunak pada sistem. Setiap bagian blok pada diagram menggambarkan suatu fungsi tertentu dari alat. Hal ini dibuat untuk memungkinkan sistem yang dibuat sudah sesuai dengan fungsi yang ada. Bentuk dari diagram blok pada sistem ini dapat dilihat pada gambar 3.7.

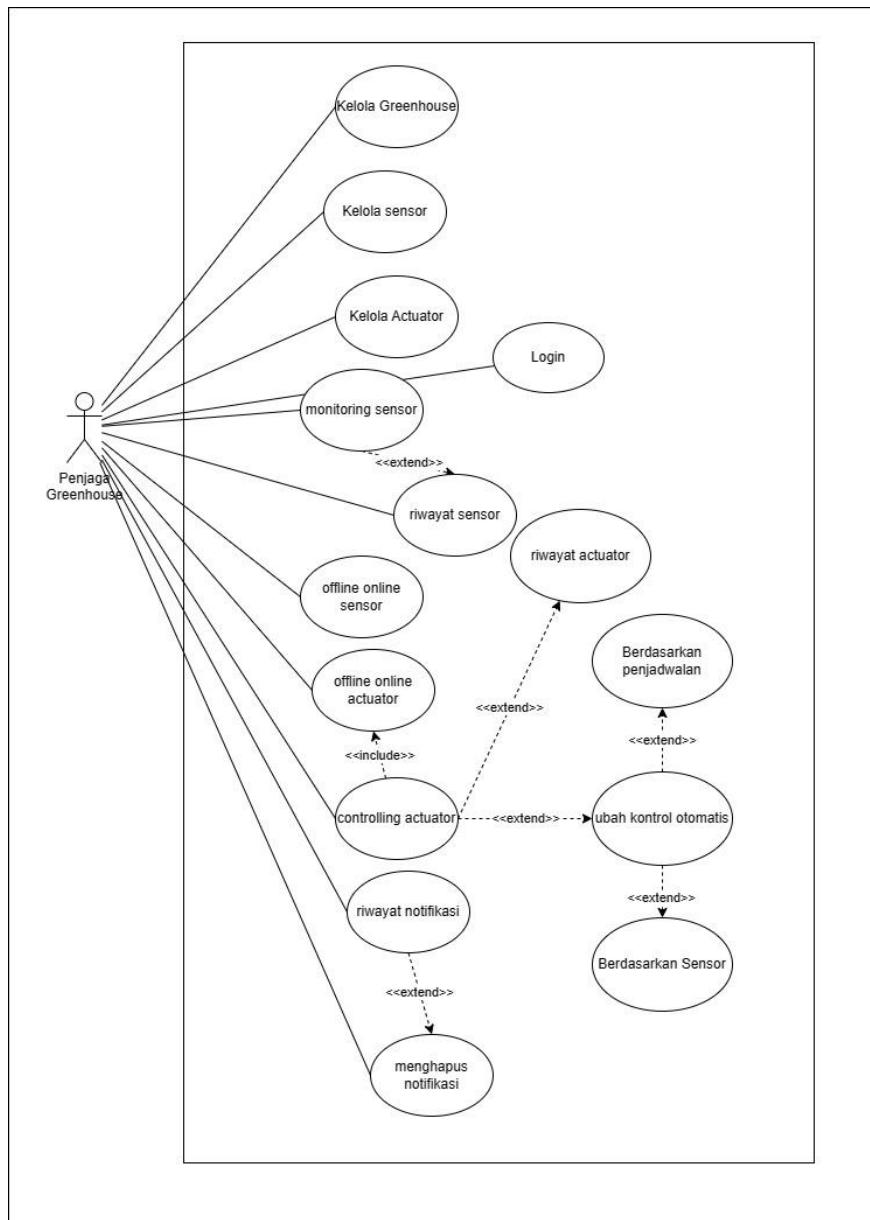


Gambar 3.7 Diagram Blok

3.4.2.2 Use Case Diagram

Diagram use case merupakan diagram yang dibuat untuk mendeskripsikan interaksi yang terjadi antara pengguna dengan sistem. Diagram ini digunakan untuk menjelaskan interaksi antara actor dan pengguna sehingga mengetahui actor yang dapat melakukan sebuah interaksi pada sistem. Diagram dapat memiliki actor yang dapat berinteraksi pada fitur-fitur

yang dimiliki sistem, tujuan dari pembuatan diagram ini adalah mendefinisikan actor dan interaksinya pada sistem. Berikut gambar diagram use case dari rancangan sistem yang akan dibangun dapat dilihat pada gambar 3.8.

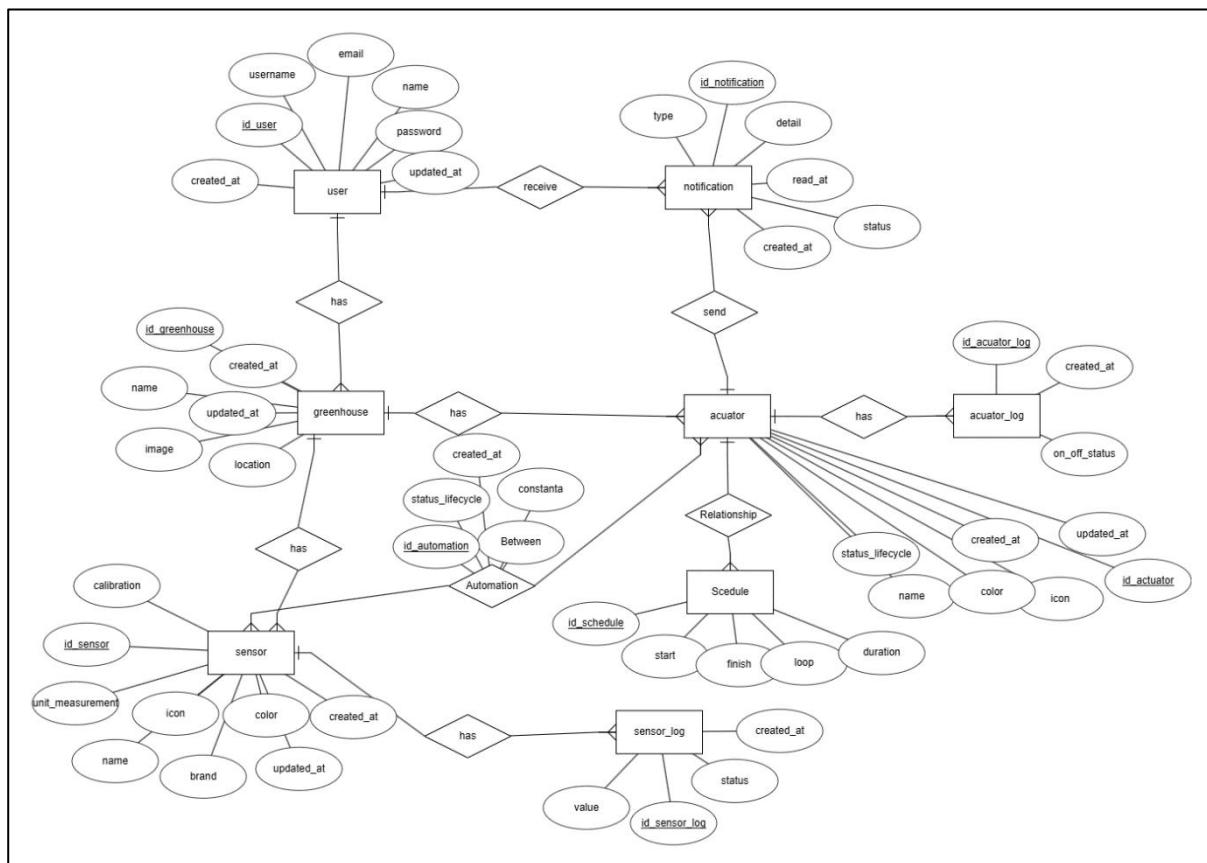


Gambar 3.8 Diagram Use Case

3.4.2.3 ER-Diagram

Entity Relationship Diagram (ERD) merupakan diagram yang dibuat untuk mendeskripsikan data-data atau objek-objek yang dibuat berdasarkan dunia nyata yang disebut entitas (entity) serta hubungan (relationship) antar entitas-entitas tersebut dengan menggunakan beberapa notasi. Manfaat dari diagram ini adalah untuk mempermudah

mendesain basis data dari objek-objek dunia nyata yang ada pada sistem. Desain ER-Diagram dari sistem monitoring *greenhouse* dapat dilihat pada gambar 3.9.



Gambar 3.11 ER Diagram

Berdasarkan gambar 3.11 didapatkan 8 entitas yakni :

1. User

User berfungsi untuk menyimpan data user sehingga user dapat login kedalam website user memiliki kardinalitas *one to many* ke notifikasi dikarenakan satu user bisa mendapatkan banyak notifikasi akan tetapi 1 notifikasi hanya dimiliki oleh 1 user saja. Lalu memiliki kardinalitas pada *greenhouse* *one to many* sehingga 1 buah user dapat memiliki banyak *greenhouse* sehingga seorang user dapat terus menambah *greenhouse*.

2. Greenhouse

Greenhouse berfungsi untuk menyimpan data *greenhouse* yang berisi nama, gambar, dan lokasi dari *greenhouse* memiliki kardinalitas *one to many* ke sensor dikarenakan satu *greenhouse* bisa memiliki banyak *greenhouse* akan tetapi 1 sensor hanya dimiliki oleh 1 *greenhouse* saja. Lalu memiliki kardinalitas pada aktuator *one to many*

sehingga 1 buah *greenhouse* dapat memiliki banyak aktuator sehingga seorang user dapat terus menambah aktuator layaknya sensor.

3. Notifikasi

Notifikasi berfungsi untuk menyimpan pemberitahuan dari user jika aktuator menyala pada jadwal tertentu sehingga memiliki kardinalitas many to one bahwa banyak notifikasi dapat dimiliki oleh satu buah aktuator

4. Aktuator

Aktuator berfungsi untuk menyimpan aktuator yang digunakan dari sebuah *greenhouse* memiliki kardinalitas one to many untuk aktuator log dikarenakan satu buah aktuator dapat memiliki banyak aktuator log yang digunakan untuk menyimpan catatan dari aktuator. Lalu memiliki kardinalitas one to many untuk scheduling dikarenakan 1 buah aktuator dapat memiliki beberapa penjadwalan.

5. Sensor

Sensor berfungsi untuk menyimpan sensor yang digunakan dari sebuah *greenhouse* memiliki kardinalitas one to many sensor log dikarenakan satu buah sensor dapat menyimpan banyak value dari sensor log. Sensor memiliki kardinalitas many to many ke aktuator dikarenakan untuk melakukan otomatisasi berdasarkan sensor 1 buah aktuator dapat memilih 1 atau lebih sensor dan juga 1 buah sensor dapat memiliki lebih dari satu buah aktuator untuk melakukan otomatisasi.

6. Sensor_log

Sensor log berfungsi untuk menyimpan nilai dari sensor saat membaca kondisi dari *greenhouse*

7. Aktuator_log

Actuator log berfungsi untuk menyimpan nilai dari actuator saat mati ataupun nyala saat di *greenhouse*

8. Schedule

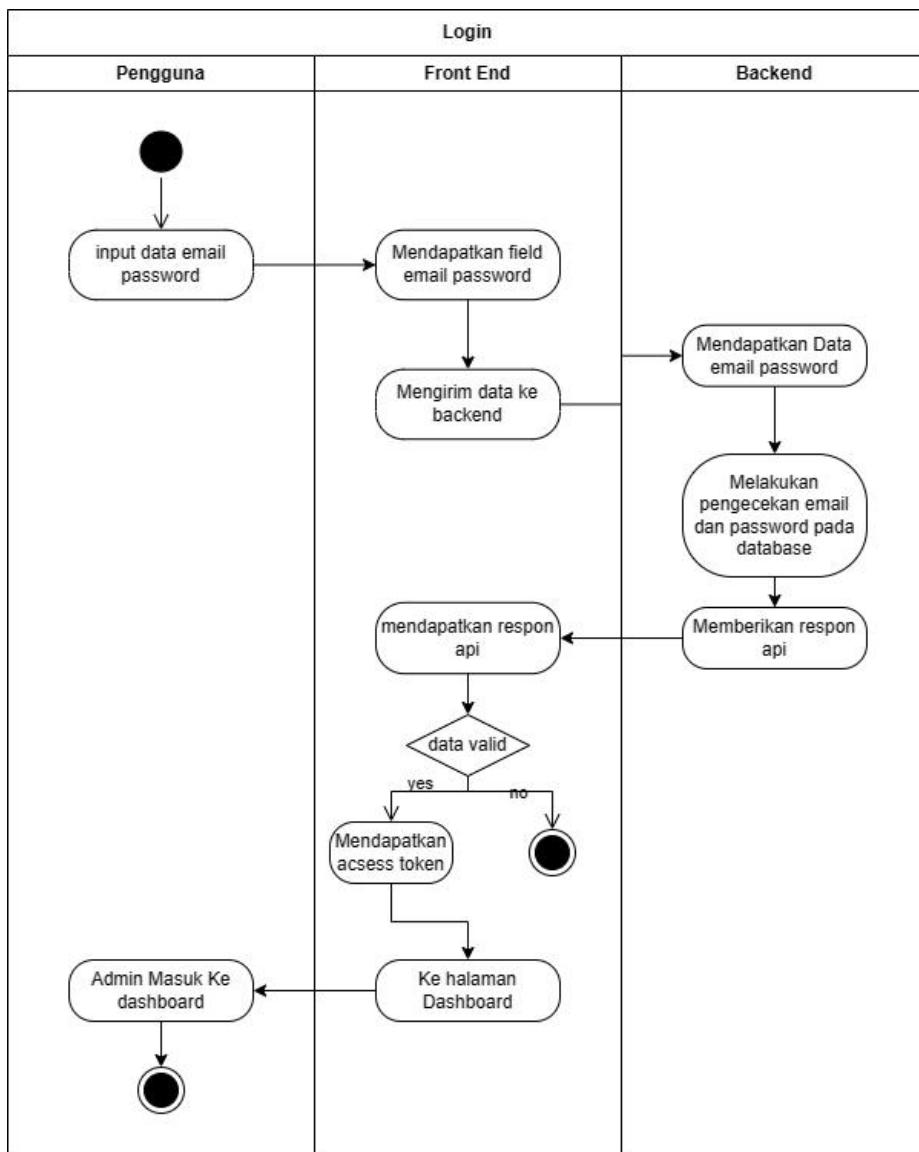
Schedule digunakan untuk menyimpan data penjadwalan dari aktuator yang dapat menyala ataupun mati secara otomatis

3.4.2.4 Activity Diagram

Activity diagram menggambarkan berbagai aliran aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing aliran berawal, decision yang mungkin terjadi dan bagaimana mereka berakhir. terdapat beberapa aktivitas yang dilakukan pada sistem monitoring *greenhouse*.

1. Tahap masuk ke sistem (login)

Pada tahapan login terdapat aktivitas yang dilakukan sehingga akhirnya dapat masuk ke dashboard. Diagram dapat dilihat pada gambar 3.9.

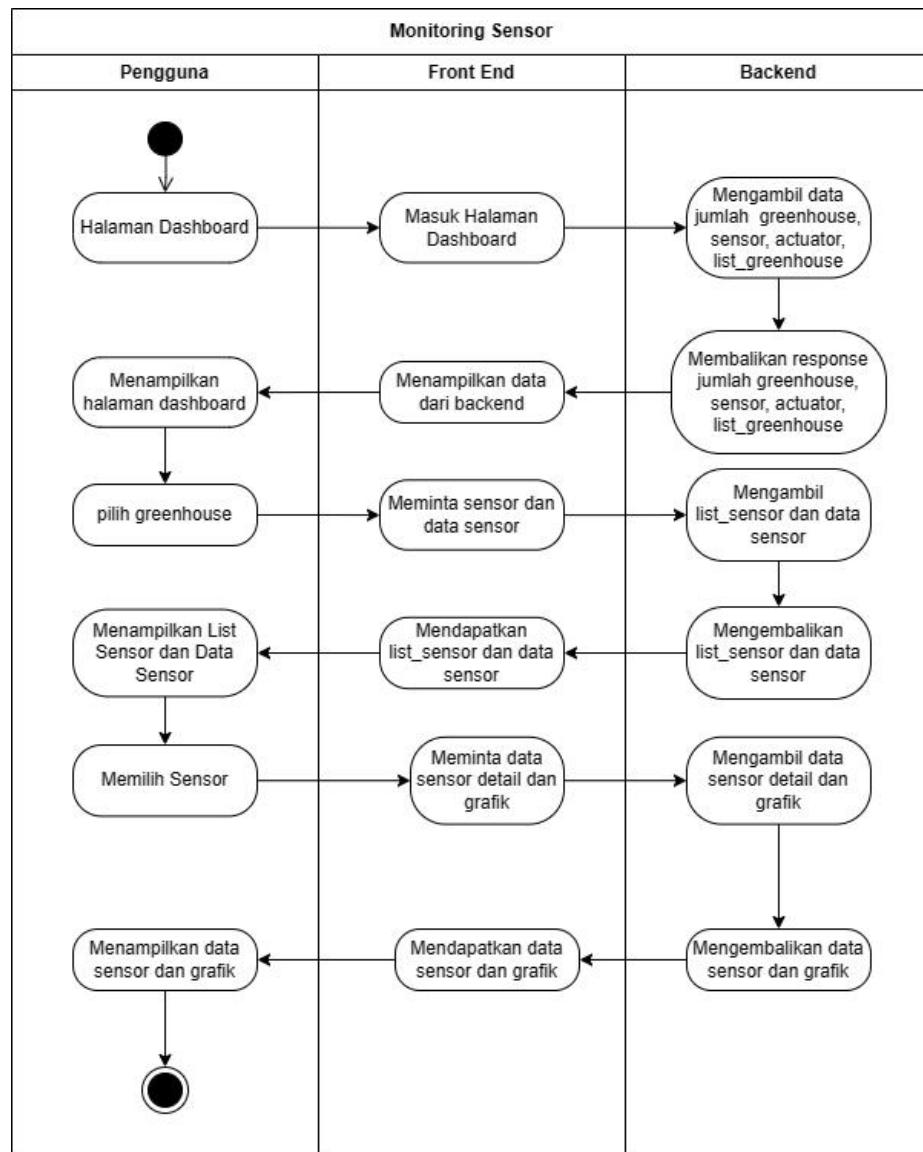


Gambar 3.9 Activity Diagram Login

2. Fitur monitoring sensor

Pada tahapan monitoring sensor terdapat merupakan fitur dimana pengguna menampilkan monitoring pada *greenhouse* sehingga dapat melihat hasil pembacaan

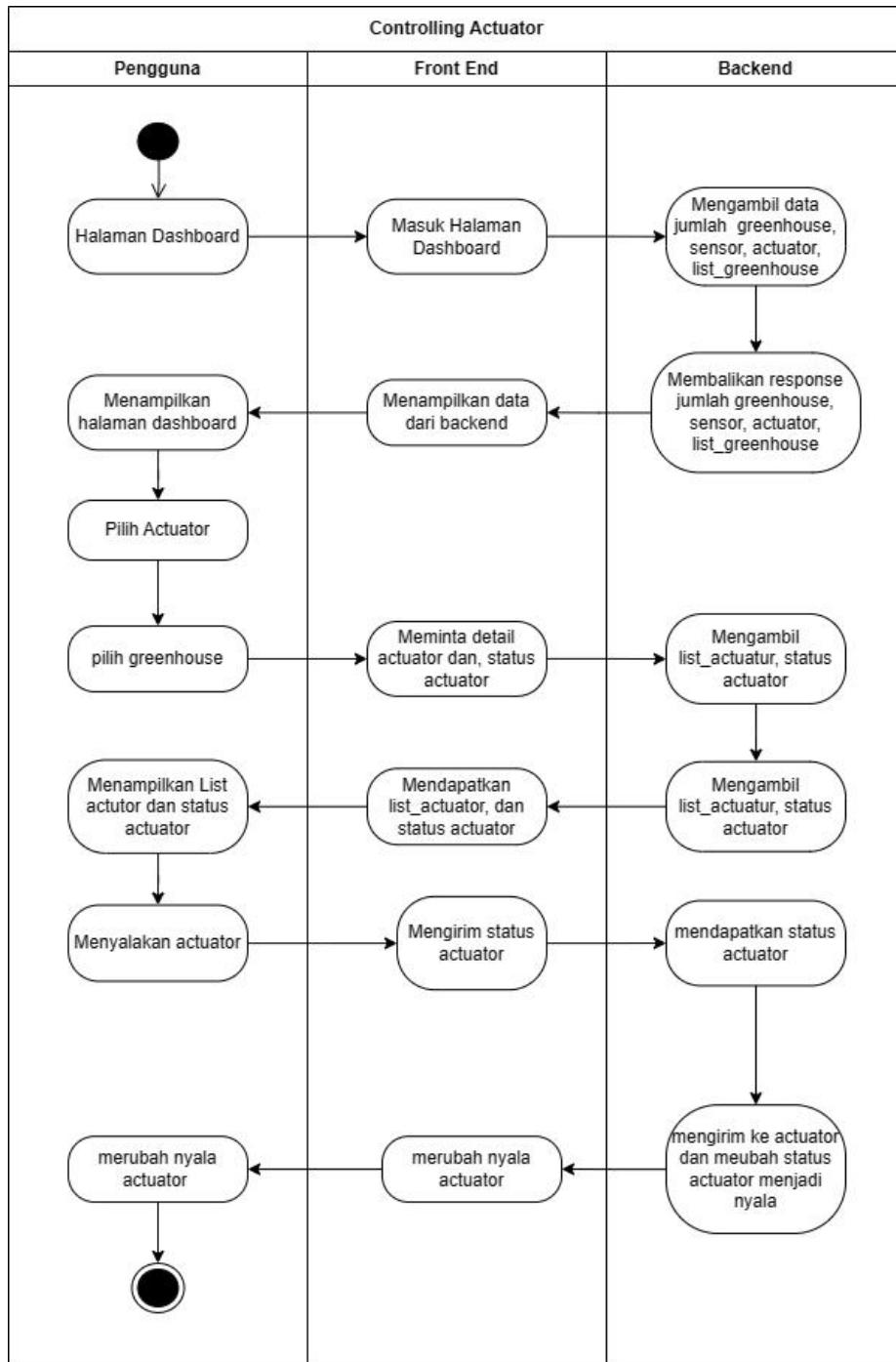
dari sensor untuk dapat melakukan monitoring terdapat tahapan-tahapan yang dapat dilihat pada table 3.10.



Gambar 3.10 Activity Diagram Monitoring Sensor

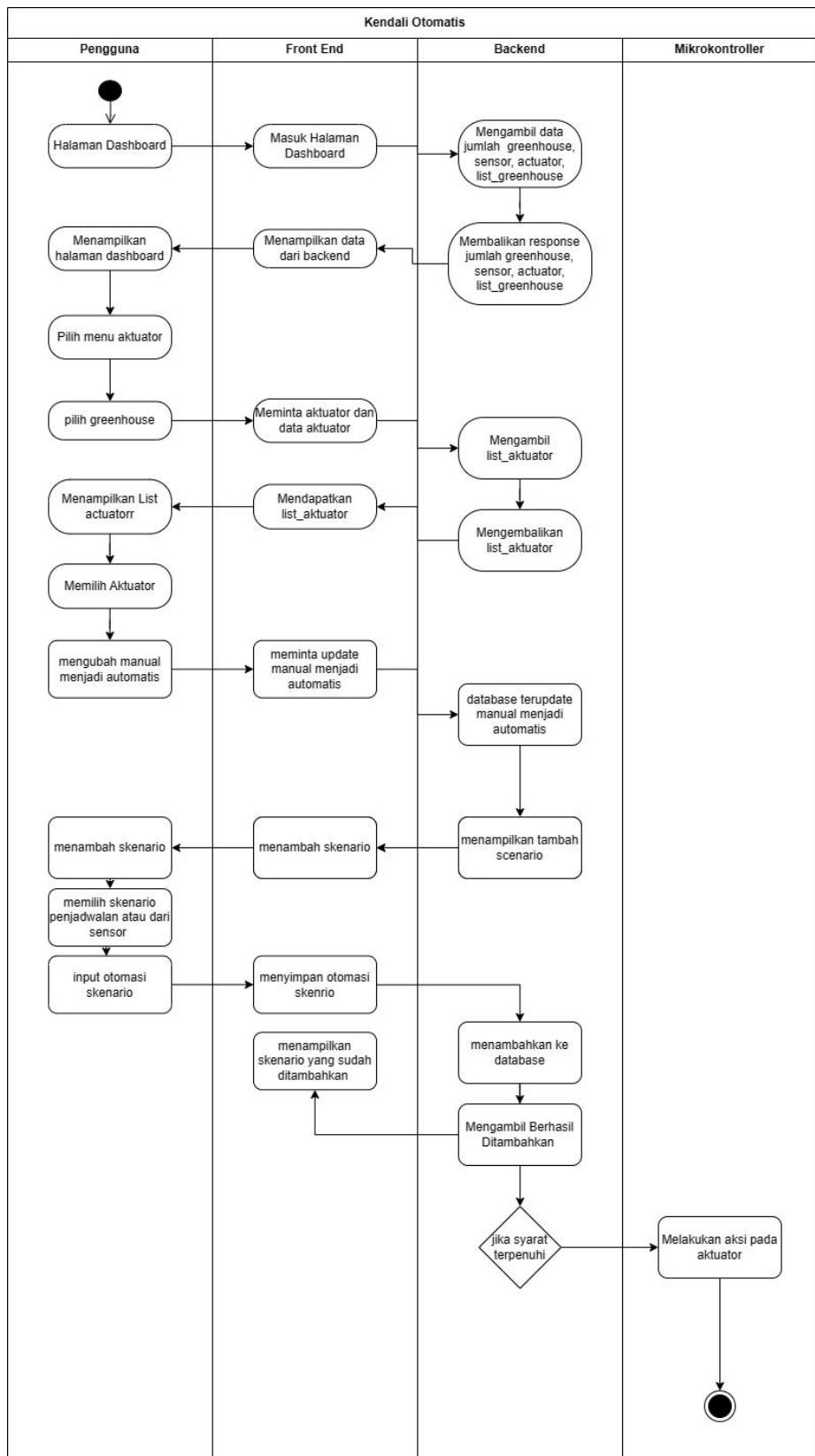
3. Fitur kontrol aktuator

Pada tahapan kontrol aktuator diperlukan proses agar dapat dikontrol oleh user untuk detail lebih jelasnya dapat dilihat pada gambar 3.11.

Gambar 3.11 Activity Diagram Kendali *Greenhouse*

4. Fitur kendali otomatis

Pada fitur kendali otomatis pada sistem ini memiliki tahapan yang perlu dilakukan, tahapan tersebut dapat dilihat pada gambar 3.12.



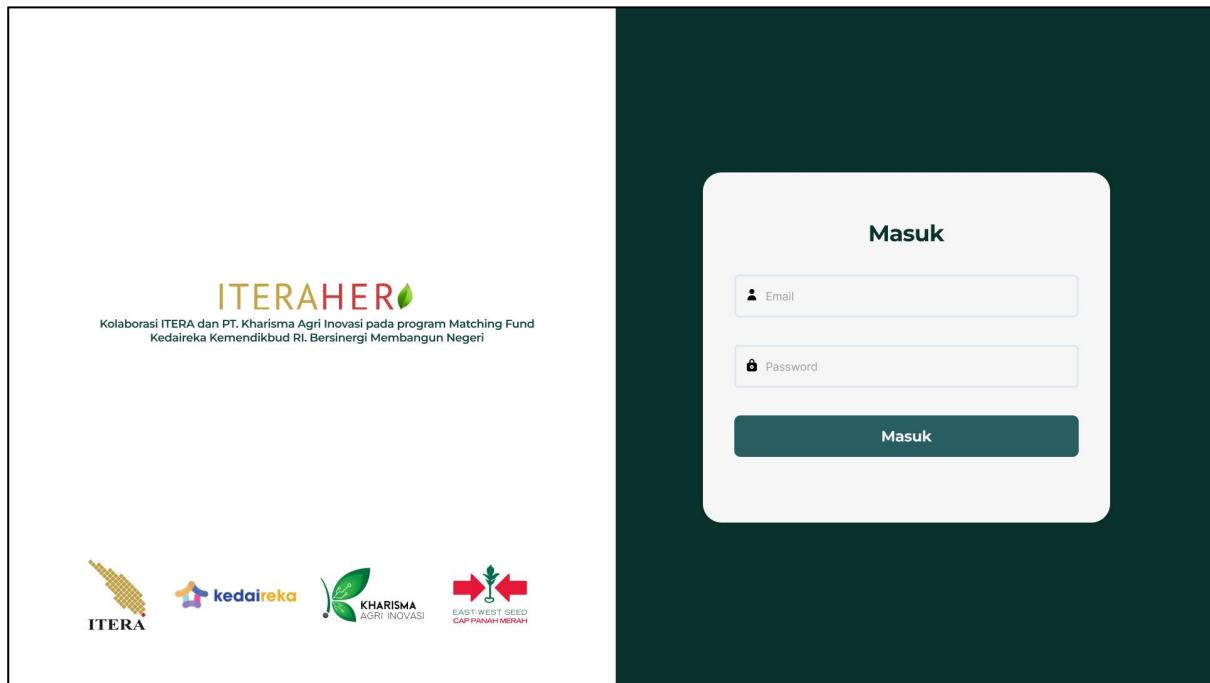
Gambar 3.12 Activity Diagram Kendali Otomatis

3.4.2.5 Design UI UX

Desain UI UX merupakan tampilan yang akan diimplementasikan pada tahap coding.

3.4.2.5.1 Login

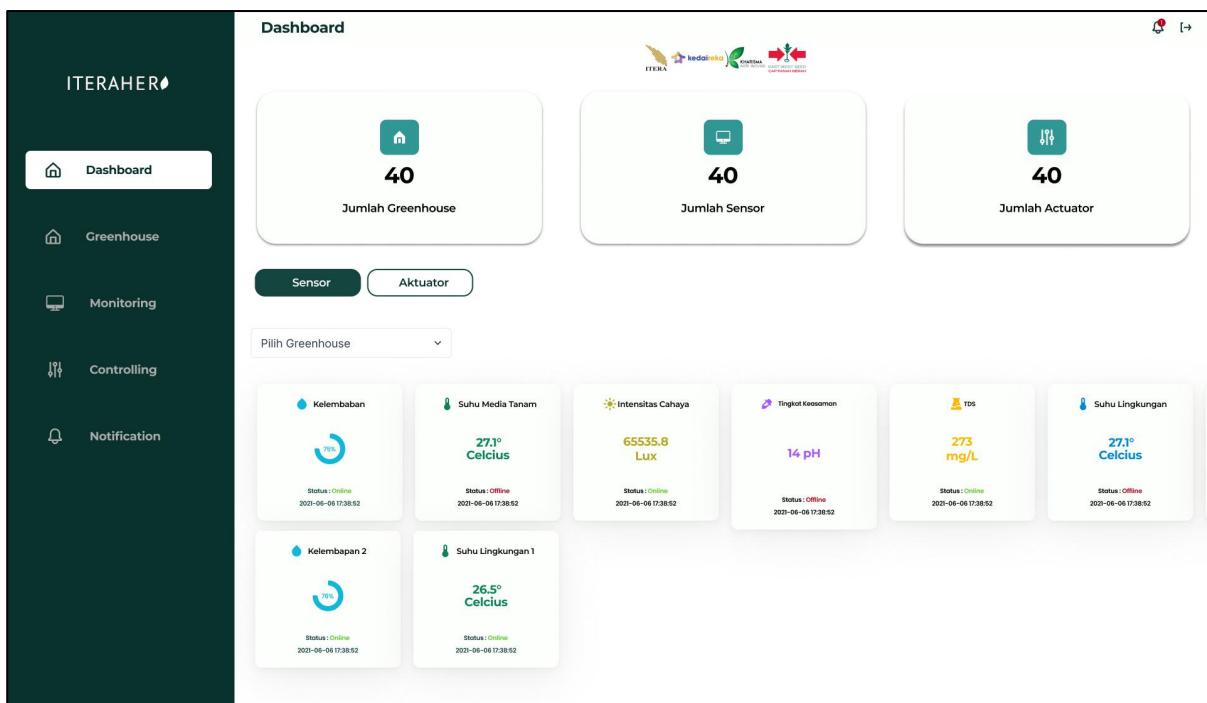
Halaman login sebagai autentikasi dari pengguna sehingga pengguna lain tidak dapat mengakses data pengguna lainnya. Pada login terdapat email dan password yang dapat diisi user agar dapat masuk kedalam sistem. Desain dapat dilihat pada gambar 3.13.



Gambar 3.13 Desain UI/UX Halaman Login

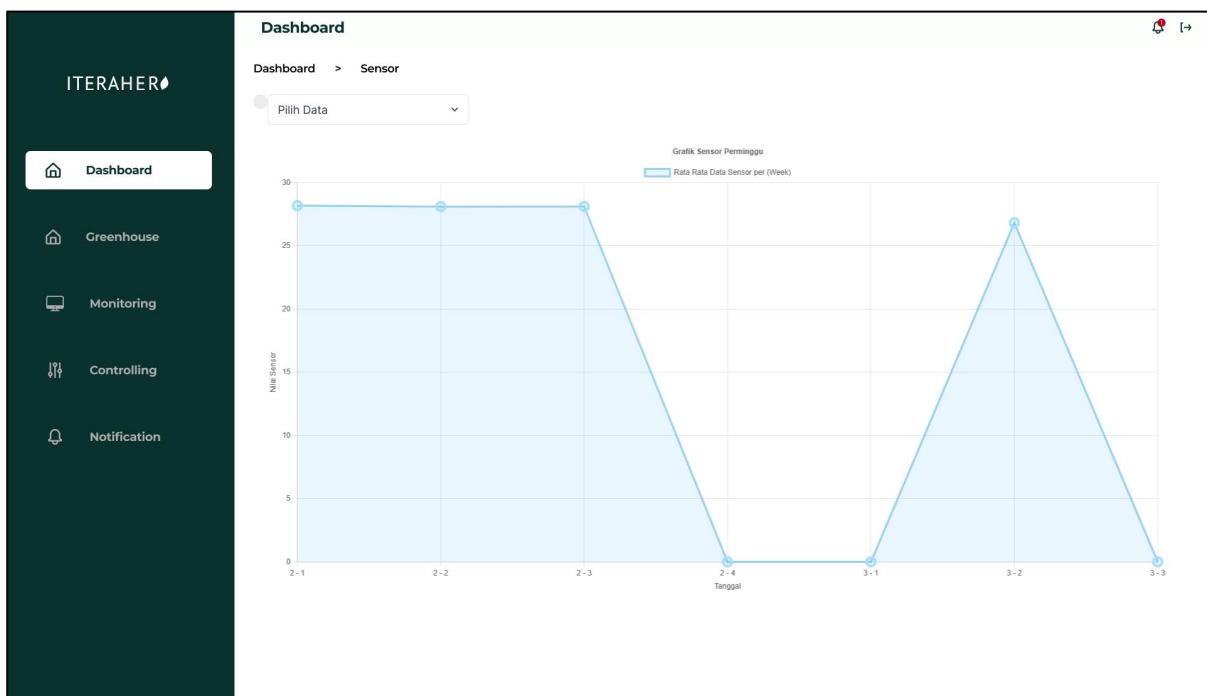
3.4.2.5.1 Halmaan Dashboard User

Setelah berhasil login akan masuk ke dalam dashboard yang berisi mengenai jumlah sensor, jumlah actuator, dan juga jumlah dari *greenhouse* yang dimiliki oleh user, selain itu pada halaman dashboard terdapat 2 menu yakni pertama dashboard untuk sensor yang berisi mengenai jumlah sensor yang berhasil dibaca. Desain tersebut dapat dilihat pada gambar 3.14.



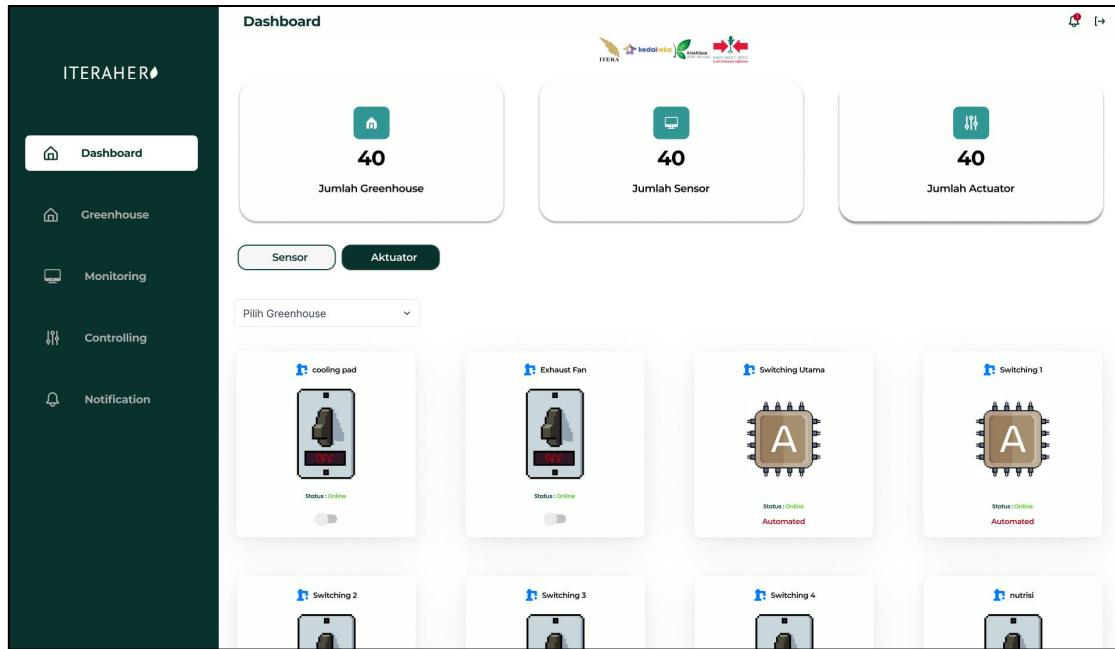
Gambar 3.14 Desain UI/UX Halaman Dashboard Sensor

Jika sensor di klik akan masuk kedalam menu grafik dapat melihat hasil dari data sensor yang telah disimpan di dalam database user dapat melihat data mingguan bulanan hingga tahunan pada website sehingga dapat melihat data historis sensor. Desain dapat dilihat pada gambar 3.15.



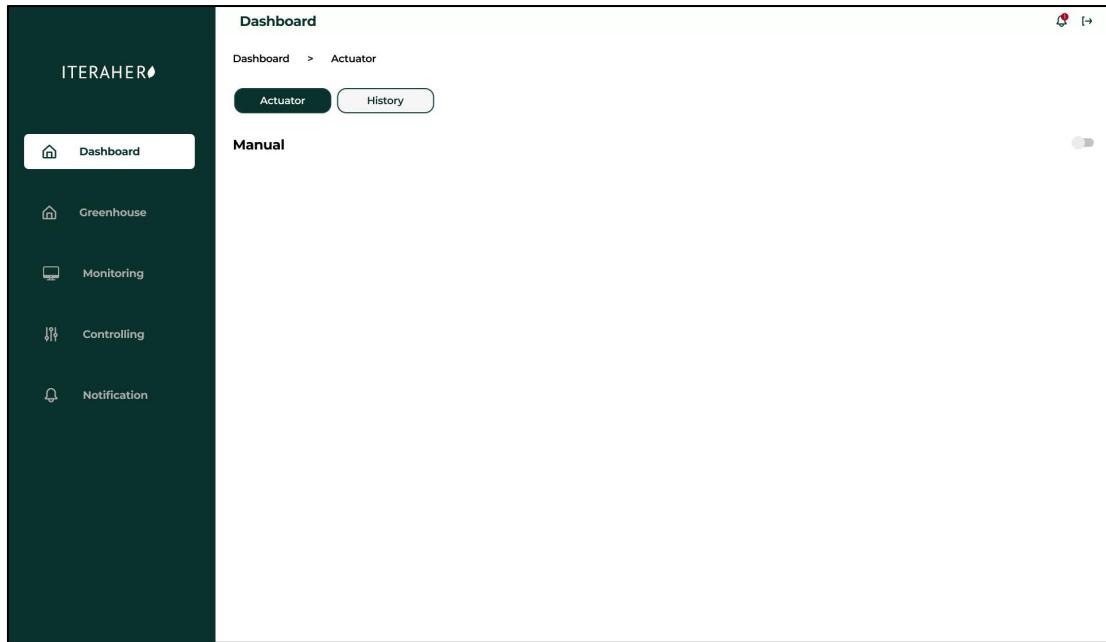
Gambar 3.15 Desain UI/UX Grafik Sensor

Selain sensor menu kedua yakni actuator yang berisi mengenai pengendalian dari *greenhouse* yakni untuk mematikan dan menyalakan kendali dari *greenhouse* jika actuator di set manual akan tetapi jika otomatis maka sistem tidak dapat dinyalakan dan dimatikan secara manual seperti pada gambar 3.16.



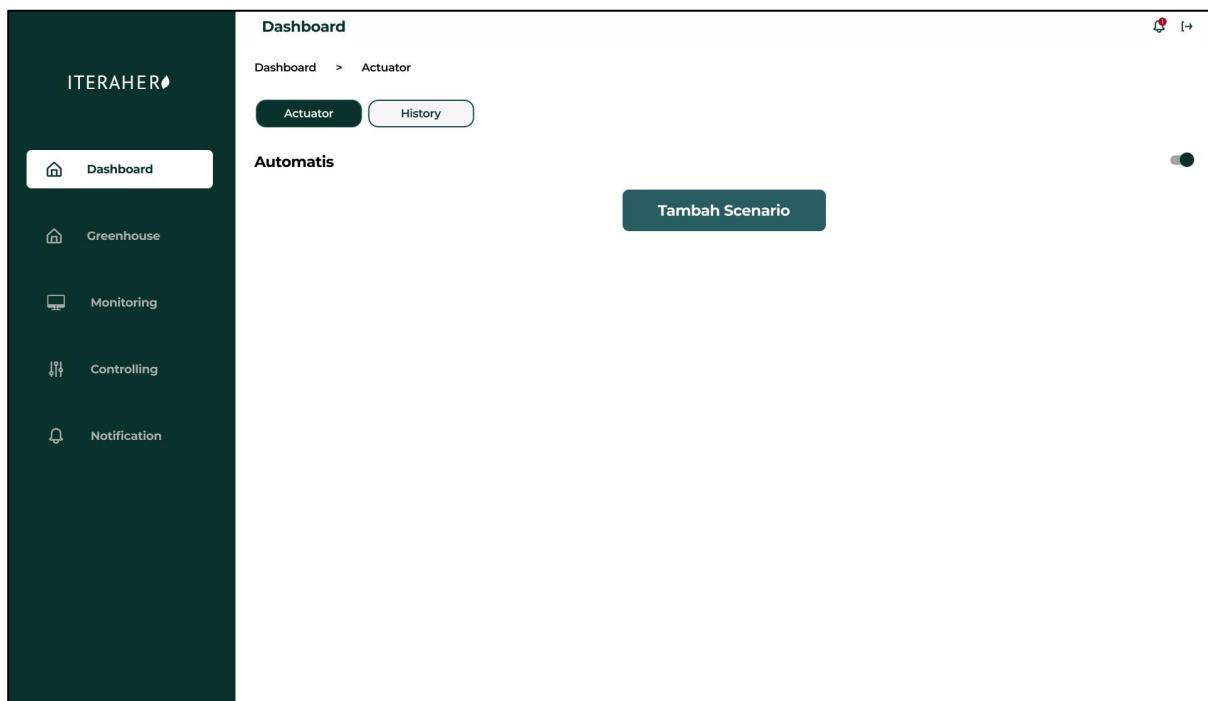
Gambar 3.16 Desain UI/UX Halaman Dashboard Actuator

jika aktuator di klik akan masuk kedalam mode untuk memilih otomatis atau manual pada menu ini kita dapat mengubah manual menjadi otomatis seperti pada gambar 3.17.



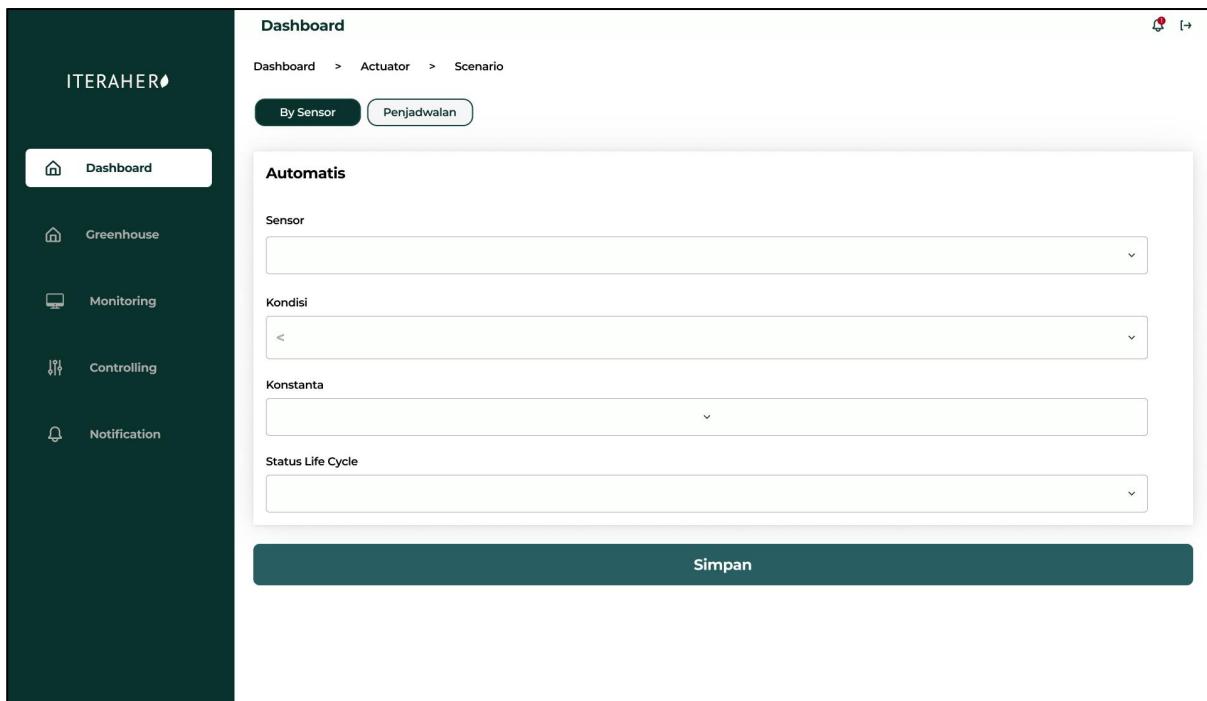
Gambar 3.17 Desain UI/UX Halaman Otomatisasi Manual

Setelah memilih otomatis maka dapat tambah skenario yakni penjadwalan atau otomatis dari sensor dengan mengekan tombol tambah scenario seperti pada gambar 3.18.



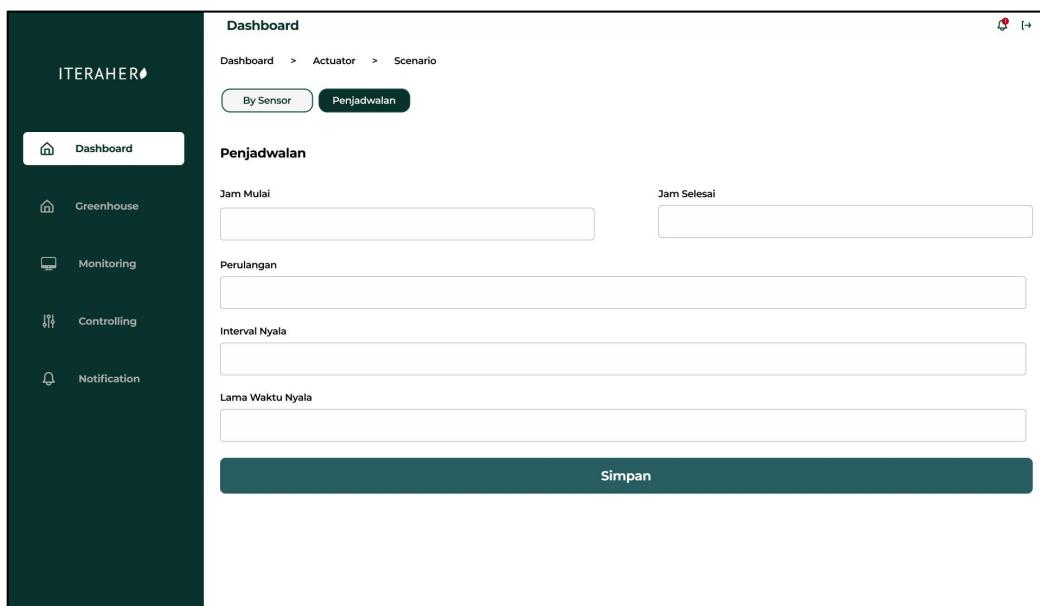
Gambar 3.18 Desain UI/UX Halaman Otomatis

saat klik tambah skenario masuk kedalam input otomatis yakni penjadwalan berdasarkan sensor. Untuk mengisi berdasarkan sensor perlu mengisi sensor, kondisi, serta konstanta dan juga status life cycle dari skenario berdasarkan sensor seperti pada gambar 3.19.



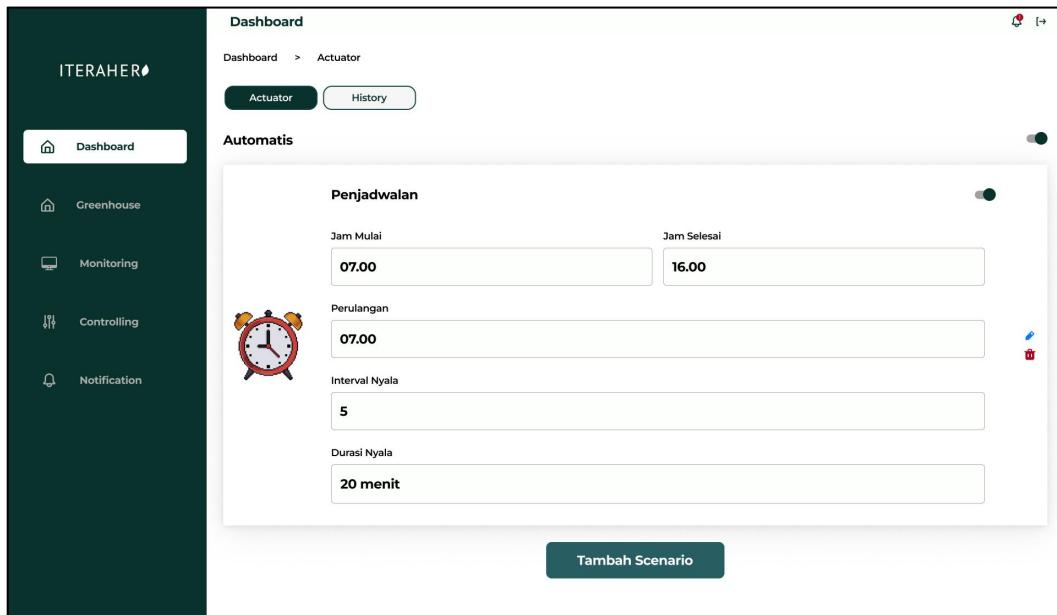
Gambar 3.19 Desain UI/UX Halaman Otomatis Tambah Skenario By Sensor

lalu selain dari sensor penjadwalan juga berdasarkan waktu dapat mengisi jam mulai dan jam selesai setelah itu perulangan digunakan untuk berapa kali sistem akan melakukan aksi berdasarkan rentan jam lalu durasi atau lama waktu nyala sehingga jika sistem sudah menyala sesuai dengan durasi maka akan mati secara otomatis seperti pada gambar 3.20.



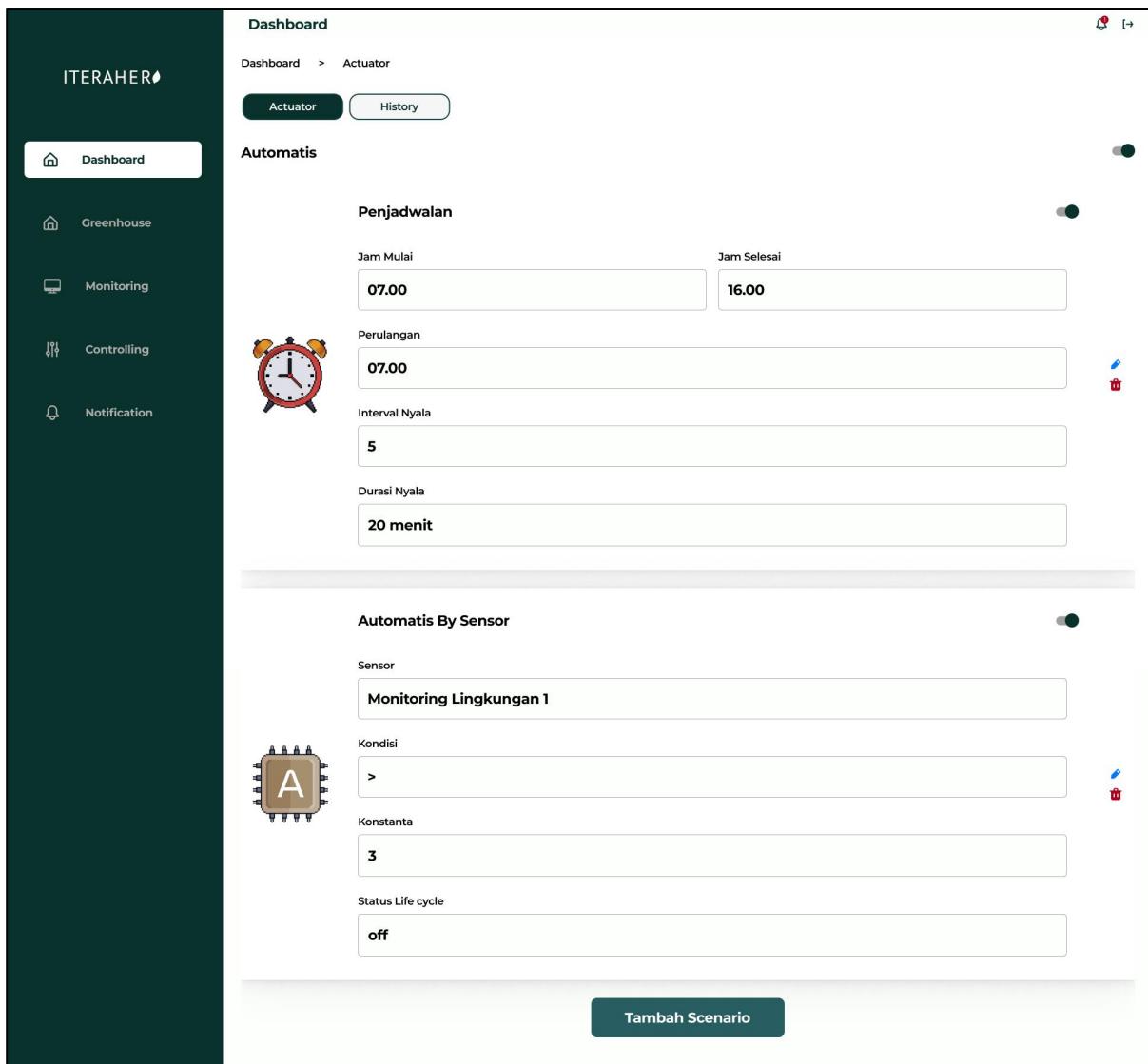
Gambar 3.20 Desain UI/UX Halaman Otomatis Tambah Skenario Penjadwalan

jika berhasil ditambahkan pada menu otomatis akan bertambah skenario yang telah di input maka akan tampil pada menu memilih otomatis dan juga manual skenario baru. Skenario dapat dinonaktifkan ataupun dinyalakan diedit serta dihapus seperti pada gambar 3.21,



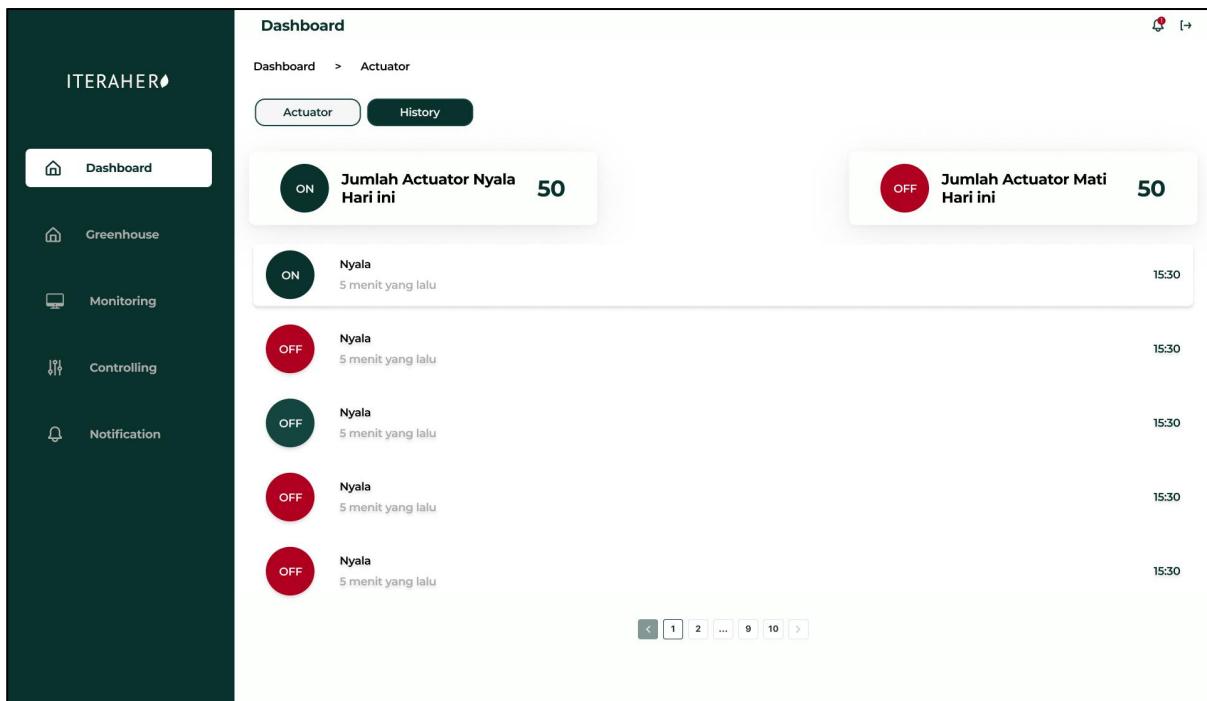
Gambar 3.21 Desain UI/UX Halaman Penambahan Skenario Penjadwalan

user dapat menambahkan skenario lebih dari 1 dan dapat mengedit mendelete ataupun mematikan skenario yang ingin dimatikan seperti pada gambar 3.22.



Gambar 3.22 Desain UI/UX Halaman Otomatis Berhasil menambahkan lebih dari satu Skenario

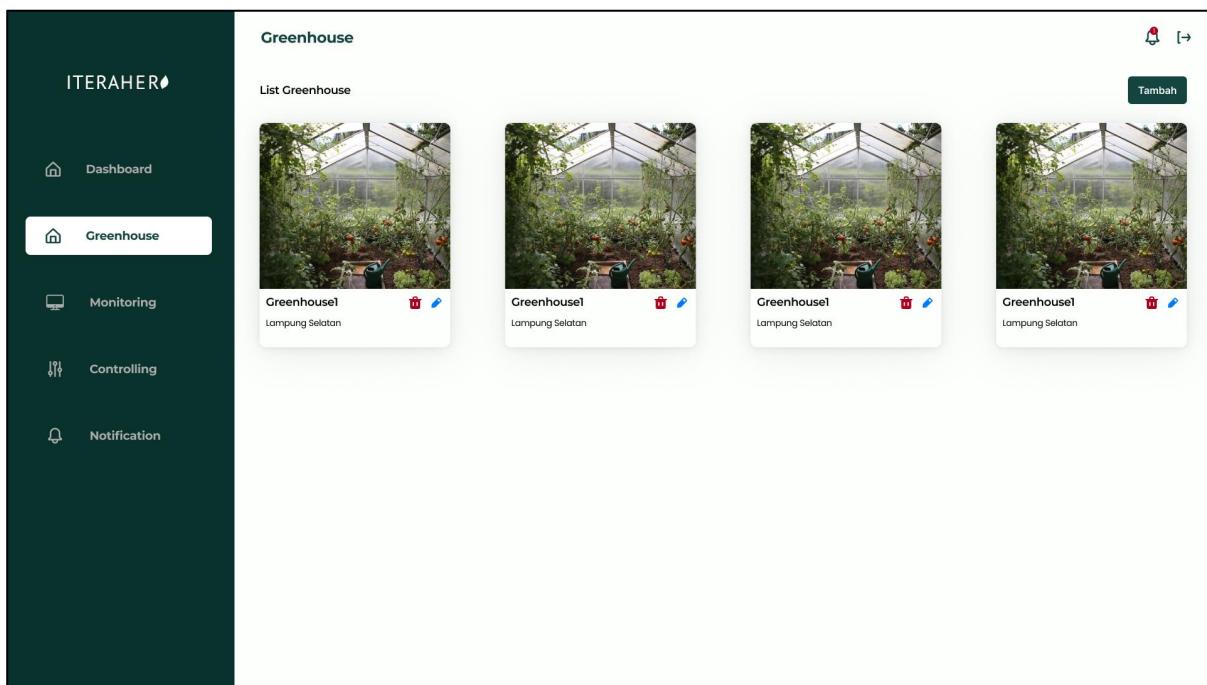
Pada menu actuator juga terdapat data historis dari aktuator yang nyala ataupun mati untuk monitoring dari kendali *greenhouse* data akan berisi mengenai kapan waktunya menyala jumlah actuator yang telah menyala hari ini dan jumlah aktuator memiliki perintah mati pada hari ini detailnya dapat dilihat pada gambar 3.23.



Gambar 3.23 Desain UI/UX Halaman Catatan Aktuator

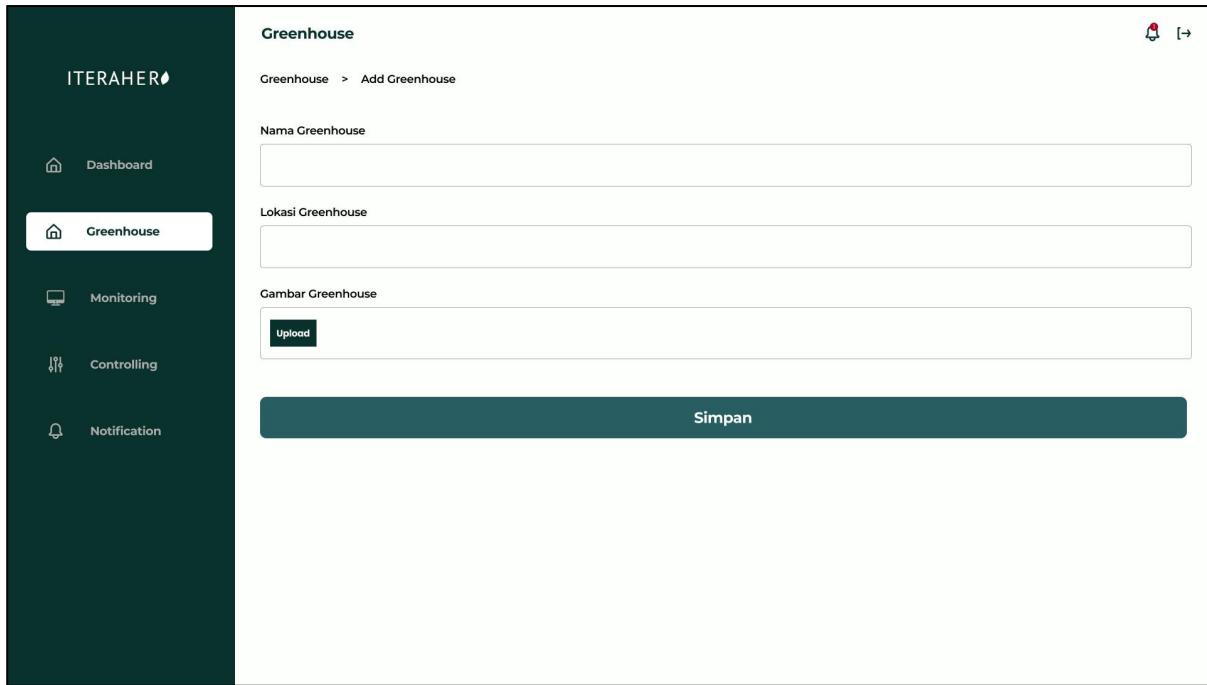
3.4.2.5.2 Halaman *Greenhouse*

pada halaman *greenhouse* berisi mengenai *greenhouse* yang dimiliki oleh user berisi mengenai nama gambar dan juga lokasi dari *greenhouse* seperti pada gambar 3.24.



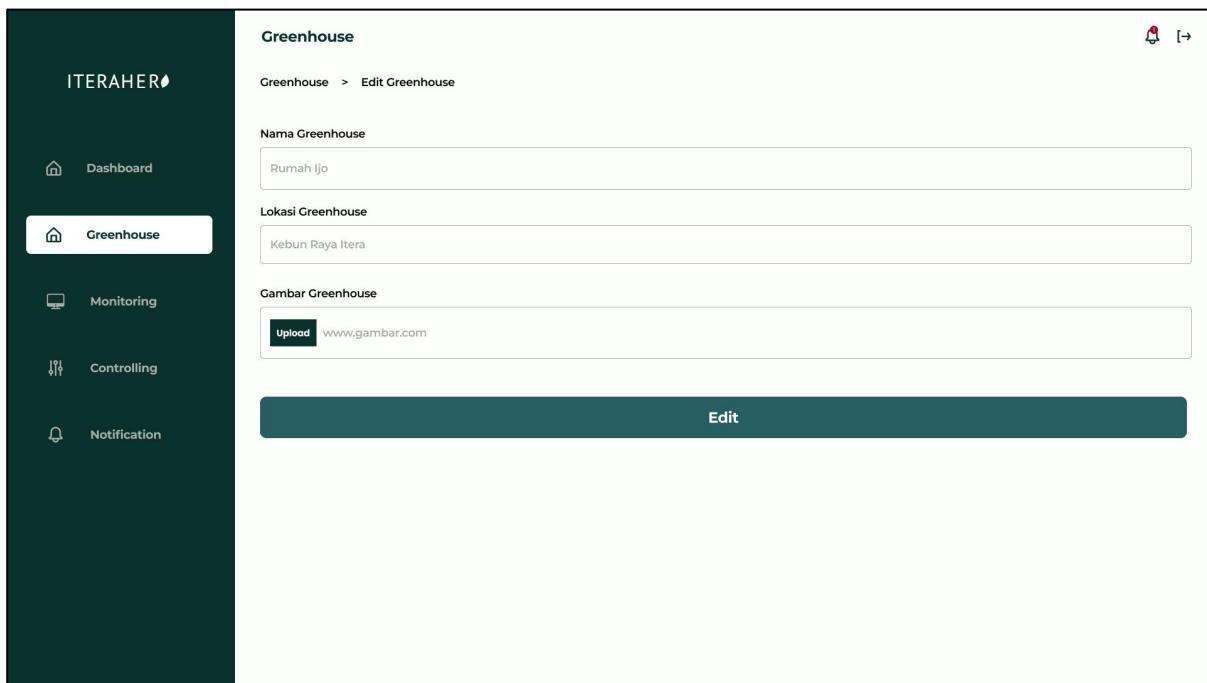
Gambar 3.24 Desain UI/UX Halaman Menu *Greenhouse*

Jika di klik tambah dapat menambah *greenhouse* menjadi lebih banyak dengan mengisikan nama, lokasi, dan juga gambar dari *greenhouse* seperti pada gambar 3.25.



Gambar 3.25 Desain UI/UX Halaman Tambah *Greenhouse*

jika terjadi kelelahan dapat melakukan edit ataupun delete pada *greenhouse* mulai dari mengganti nama lokasi ataupun gambar jika tidak ingin diubah maka data diisikan seperti sebelumnya, desain dapat dilihat pada gambar 3.26.



Gambar 3.26 Desain UI/UX Halaman Edit *Greenhouse*

3.4.2.5.3 Halaman Sensor

Setelah membuat *greenhouse* kita dapat menambahkan sensor berdasarkan *greenhouse* yang dimiliki. Sensor dapat dihubungkan dengan sistem tertanam yang telah coding lalu akan muncul di dashboard seperti pada gambar 3.27.

| Nomor | Ikon | Satuan Ukur | Nama | Merek | Kode Warna | Aksi |
|-------|------|-------------|----------|-------|------------|------|
| 1 | ☀️ | Lux | DXDIAG | Fruit | Kuning | |
| 2 | ☀️ | Lux | DXDIAG | Fruit | Kuning | |
| 3 | ☀️ | Lux | DXDIAG | Fruit | Kuning | |
| 4 | ☀️ | Lux | DXDIAG | Fruit | Kuning | |
| 5 | ☀️ | Lux | DXDIAG | Fruit | Kuning | |
| 6 | ☀️ | Lux | DXDIAG | Fruit | Kuning | |
| 7 | ☀️ | Lux | DXDIAG | Fruit | Kuning | |
| 8 | ☀️ | Lux | SSIS-187 | Fruit | Kuning | |
| 9 | ☀️ | Lux | SSIS-187 | Fruit | Kuning | |
| 10 | ☀️ | Lux | SSIS-187 | Fruit | Kuning | |

Gambar 3.27 Desain UI/UX Halaman Menu Monitoring

Jika klik tambah akan dapat menambahkan sensor baru yang nantinya dapat dihubungkan dengan sensor yang telah dibuat menggunakan mikrokontroler mulai dari nama ikon satuan ukur merek dan juga kategori sensor seperti pada gambar 3.28.

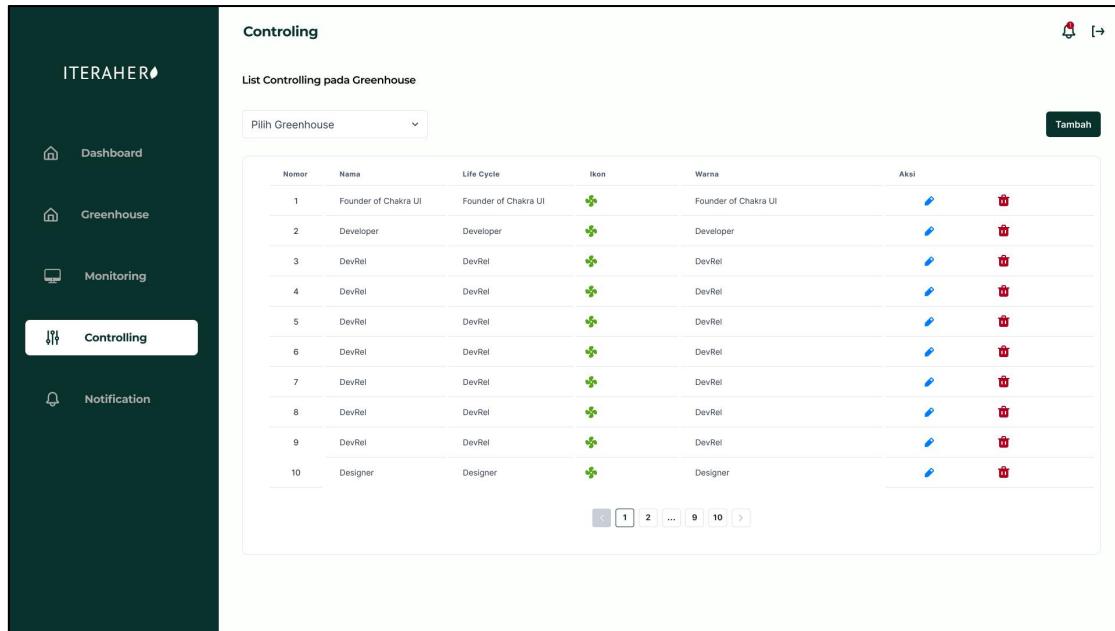
Gambar 3.28 Desain UI/UX Halaman Edit *Greenhouse*

Jika terdapat kesalahan pada sensor dapat mengubah sensor dengan edit ataupun menghapus sensor dengan mengedit seperti ikon satuan ukur nama ataupun merek dari sensor seperti pada gambar 3.29.

Gambar 3.29 Desain UI/UX Halaman Edit Monitoring

3.4.2.5.4 Halaman Kendali

Setelah membuat *greenhouse* kita dapat menambahkan aktuator berdasarkan *greenhouse* yang dimiliki. Aktuator dapat dihubungkan dengan sistem tertanam yang telah *coding* lalu akan muncul di *dashboard*. Desain dapat dilihat pada gambar 3.30.

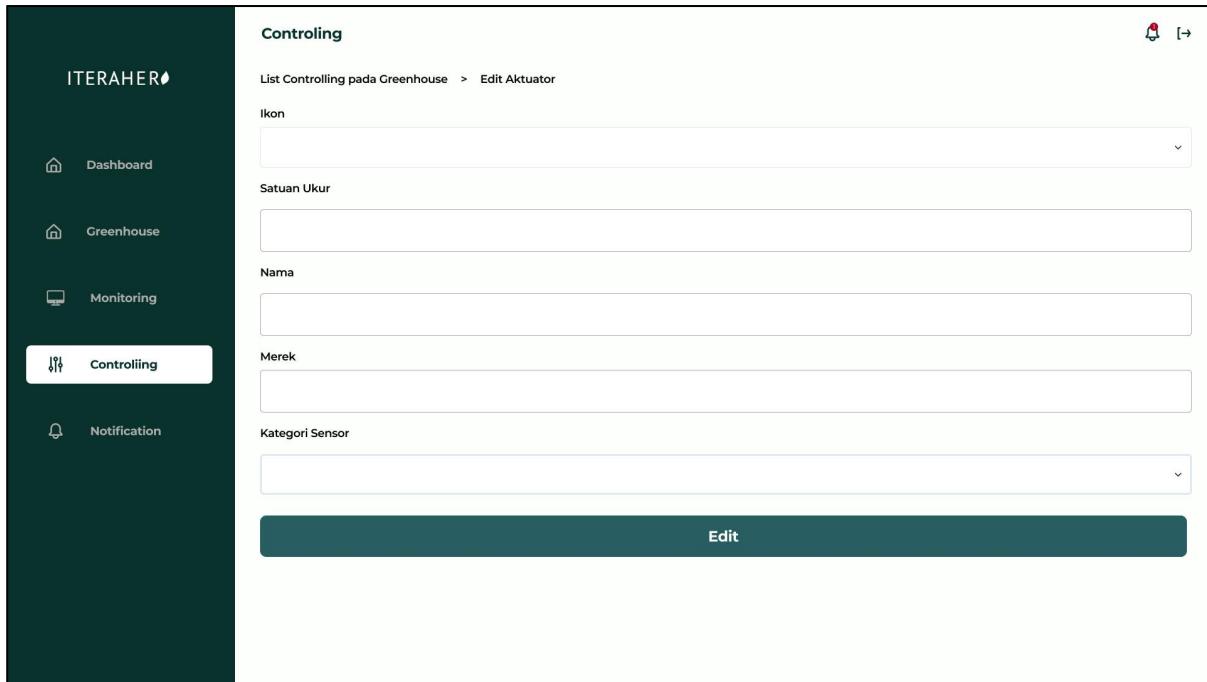


Gambar 3.30 Desain UI/UX Halaman Menu Actuator

Jika klik tambah akan dapat menambahkan sensor baru yang nantinya dapat dihubungkan dengan sensor yang telah dibuat menggunakan mikrokontroler. Desain dapat dilihat pada gambar 3.31.

Gambar 3.31 Desain UI/UX Halaman Tambah Actuator

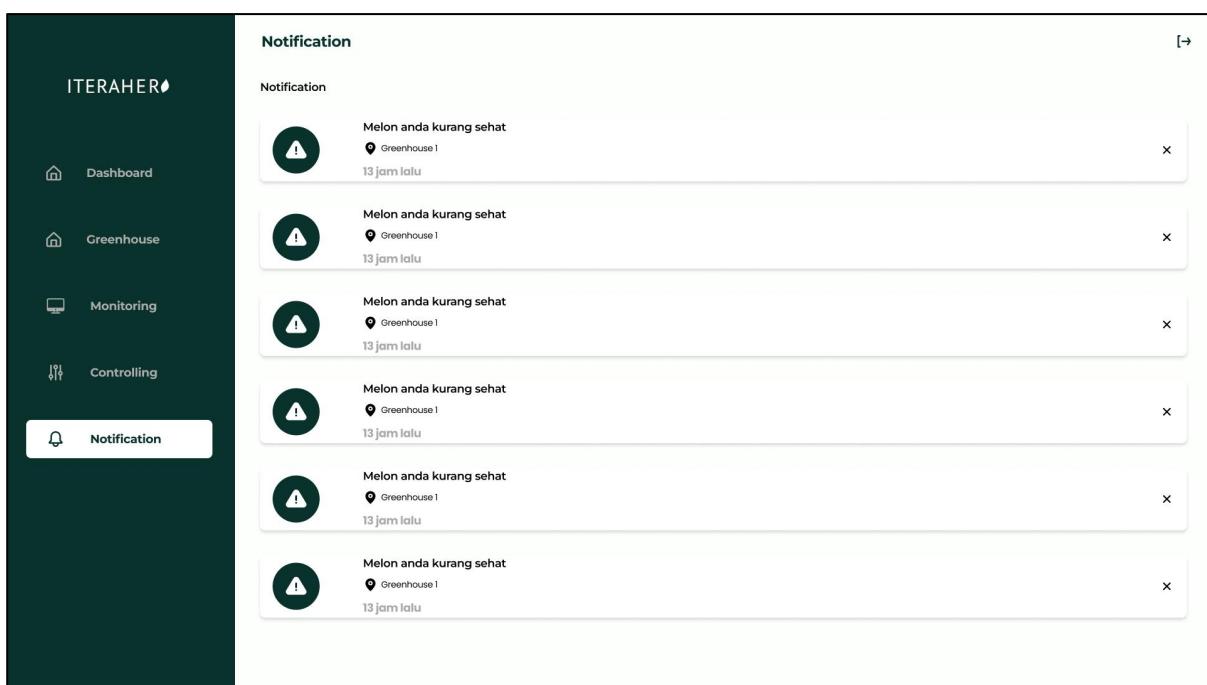
Jika terdapat kesalahan pada aktuator dapat mengubah sensor dengan edit ataupun menghapus sensor yang dapat dilihat pada gambar 3.32.



Gambar 3.32 Desain UI/UX Halaman Edit Aktuator

3.4.2.5.5 Halaman Notifikasi

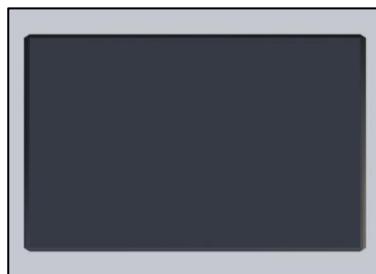
Halaman notifikasi berisi mengenai notifikasi yang muncul di sistem jika terjadi masalah pada alat jika sudah mencapai batas maksimal ataupun minimal dapat dilihat pada gambar 3.33.



Gambar 3.33 Desain UI/UX Halaman Notifikasi

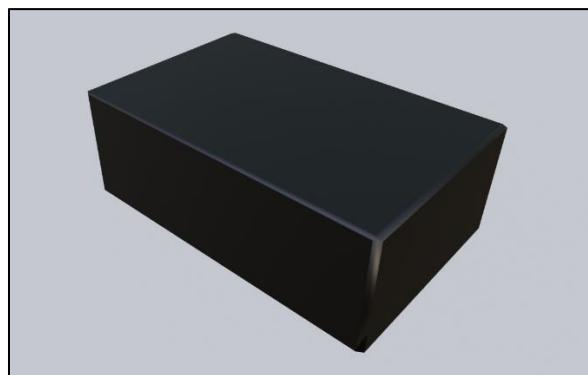
3.4.2.7 Desain *Cashing*

Dalam pembuatan alat diperlukan desain dari alat yang akan dibuat untuk alat sensor suhu, exhaust fan, cooling pad, dirancang menggunakan box berwarna hitam untuk rincian dari *cashing* yang dibuat dapat dilihat pada gambar 3.34.



Gambar 3.34 Rancangan *Cashing* box hitam tampak atas

Tampilan box hitam dapat dilihat dari samping pada gambar 3.35.



Gambar 3.35 Rancangan *Cashing* box hitam tampak samping

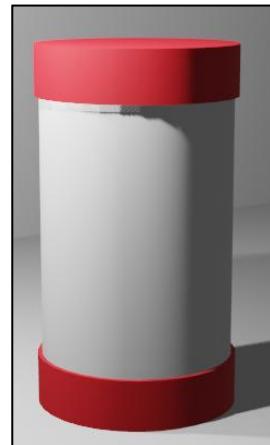
Tampilan box hitam dapat dilihat dari bawah dapat dilihat pada gambar 3.36



Gambar 3.36 *Cashing* box hitam tampak bawah

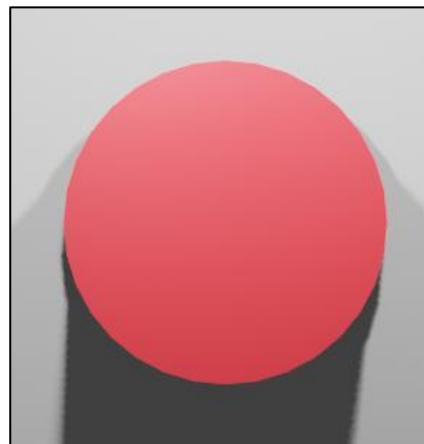
Sedangkan pada sensor debit dan seleoid valve memiliki rancangan berbentuk pipa yang nantinya akan dimasukkan *microcontroller* dan juga komponen yang terkait pada *sensor*

debit dan juga *solenoid valve* tampilan dari untuk pengendalian nutrisi dapat dilihat pada gambar 3.37.



Gambar 3.37 Rancangan *Cashing box* tampak bawah pada pengendalian nutrisi

Tampak atas rancangan dari *cashing* pengendalian dapat dilihat pada gambar 3.38.



Gambar 3.38 Rancangan *Cashing box* tampak atas pada pengendalian nutrisi

3.4.3 Develop

Pada tahapan ini dilakukan proses pengimplementasian pembuatan sistem dengan membuat program sesuai dengan kebutuhan fungsional yang didapatkan pada tahapan sebelumnya. Pembuatan program yang akan diimplementasikan terlebih dahulu berdasarkan kebutuhan fungsional.

3.4.4 Testing (Pengujian)

Pada tahapan ini dilakukan pengujian terhadap sistem yang dibangun dengan menggunakan Black box testing. Pengujian ini dilakukan terhadap program yang telah dibuat sebelumnya

untuk mengetahui apakah fungsi/fitur yang dibangun sesuai dengan user story. Rancangan pengujian lebih lengkap dapat dilihat pada subbab 3.5.

3.4.4 Deploy

Pada tahapan ini dilakukan hosting ke server sehingga dapat dilakukan tahap pengujian kepada klien mengenai sistem yang dibuat.

3.4.4 Review Klien

Pada tahapan ini dilakukan pengujian terhadap sistem yang dibangun dengan menggunakan SUS (*System Usability Scale*) untuk mengevaluasi sistem apakah sudah sesuai dengan kebutuhan yang ada dan dapat digunakan dengan baik.

3.4.4 Launch

Pada tahapan ini dilakukan tahap penyelesaian atau penutupan dari proyek untuk selesai dikembangkan dan siap digunakan di *greenhouse* dan pada tahap ini terdapat hasil dari pemasangan alat yang telah dibuat dan membandingkan hasil antara *smart greenhouse* dan peneanaman dengan menggunakan metode konvensional, untuk perbandingan yang dilakukan perbandinan melon mulai dari berat, tiskat kemanisan, dan juga ukuran dari melon.

3.5 Rancangan Pengujian

Pengujian bertujuan untuk mengetahui kelayakan sistem yang dibangun untuk digunakan oleh pengguna. Pada sistem yang akan dibangun akan dilakukan pengujian dengan menggunakan metode black box testing. Metode pengujian ini akan menguji setiap fitur yang ada pada sistem dengan indikator keberhasilan berupa fungsi dari setiap fitur pada sistem yang dibangun sesuai dengan user story. Rancangan pengujian dapat dilihat pada tabel 3.11.

Tabel 3.11 Rancangan Pengujian Black box

| Fitur | Case | Indikator Keberhasilan |
|-----------|--|---|
| Login | Menginput sesuai dengan email dan password | Masuk ke halaman home |
| | Menginput tidak sesuai dengan email dan password | Kembali ke login dengan pemberitahuan login gagal |
| Dashboard | Masuk ke halaman dashboard | Menampilkan jumlah sensor, actuator dan <i>greenhouse</i> |

| Fitur | Case | Indikator Keberhasilan |
|-------------------------------|--|--|
| Menambah <i>greenhouse</i> | Menginput <i>greenhouse</i> dengan data lengkap | Berhasil menambah <i>greenhouse</i> |
| | Menginput <i>greenhouse</i> dengan data yang tidak lengkap | tidak dapat di submit |
| Delete <i>greenhouse</i> | Menghapus <i>Greenhouse</i> | <i>Greenhouse</i> terhapus |
| Edit <i>greenhouse</i> | Mengedit <i>greenhouse</i> | <i>Greenhouse</i> teredit |
| Menampilkan <i>greenhouse</i> | ke halaman <i>greenhouse</i> dan menampilkan <i>greenhouse</i> | <i>Greenhouse</i> tampil |
| Menambah Aktuator | Menginput aktuator dengan data lengkap | Berhasil menambah <i>greenhouse</i> |
| Delete aktuator | Menginput aktuator dengan data yang tidak lengkap | tidak dapat di submit |
| Delete Aktuator | Menghapus aktuator | aktuator terhapus |
| Edit Aktuator | Mengedit aktuator | aktuator teredit |
| Menampilkan aktuator | Masuk ke halaman aktuator dan menampilkan aktuator | List Aktuator tampil |
| Menambah Sensor | Menginput sensor dengan data lengkap | Berhasil menambah sensor |
| | Menginput sensor dengan data yang tidak lengkap | tidak dapat di submit |
| Delete sensor | Menghapus sensor | aktuator terhapus |
| Edit sensor | Mengedit sensor | aktuator teredit |
| Menampilkan sensor | Masuk ke halaman sensor dan menampilkan sensor | List sensor tampil |
| Menampilkan Nilai Sensor | Menampilkan sensor pada halaman home | Berhasil menampilkan sensor dan data sensor yang telah dibaca dari perangkat keras |
| Menampilkan Grafik Sensor | Menampilkan grafik pada detail sensor | Berhasil menampilkan grafik sensor |
| Status sensor | Menampilkan status dari sensor | Berhasil menampilkan status sesuai dengan perangkat |

| Fitur | Case | Indikator Keberhasilan |
|--|--|---|
| | | keras |
| Status aktuator | Sistem mampu menampilkan aktuator offline dan online | Berhasil menampilkan status sesuai dengan perangkat keras |
| Sistem dapat mematikan dan menyalakan aktuator | Sistem menyalakan aktuator dan pada perangkat keras menyala | Sistem berhasil menyalakan aktuator dan pada perangkat keras menyala |
| Sistem memberikan notifikasi | menampilkan halaman notifikasi | Sistem berhasil mendapatkan notifikasi |
| Sistem mampu menghapus notifikasi | menghapus notifikasi | notifikasi terhapus |
| kontrol otomatis | Mengubah kontrol manual jadi otomatis | Sistem berhasil mengubah kontrol manual jadi otomatis |
| | | sistem berhasil menampilkan kontrol otomatis |
| kontrol dari sensor | Menambahkan aktuator secara otomatis berdasarkan sensor | Sistem berhasil Menambahkan aktuator secara otomatis berdasarkan sensor dan perangkat keras berhasil melakukan aksi |
| kontrol dari penjadwalan | Menambahkan aktuator secara otomatis berdasarkan penjadwalan | Sistem berhasil Menambahkan aktuator secara otomatis berdasarkan waktu dan perangkat keras melakukan aksi yang sesuai dari otomatis |
| riwayat aktuator | menampilkan riwayat aktuator hidup ataupun mati | Sistem mampu menampilkan riwayat aktuator hidup ataupun mati |

Selain itu terdapat pengujian perangkat keras jika sistem sudah selesai yakni dapat dilihat pada tabel 3.12.

Tabel 3.12 Pengujian Perangkat Keras

| Sensor - aktuator | Case | Indikator Keberhasilan |
|-------------------|--|---|
| Suhu – Kipas | Akan menyala jika suhu mencapai 30 derajat Celcius dan mati pada suhu 28 | Kipas dan Cooling Pad Menyala dan mati sesuai kondisi |

| | | |
|-------------------------------|---|---|
| | derajat celcius | |
| Sensor Debit – Selenoid Valve | Akan menyala sesuai dengan perintah secara otomatis dan mati sesuai dengan debit 5 ml | Selenoid Valve berhasil menyala dan mati sesuai kondisi |
| Kelembaban - Cooling Pad | Akan menyala saat kelembaban kurang dari 70% | Cooling pad menyala sesuai kondisi |

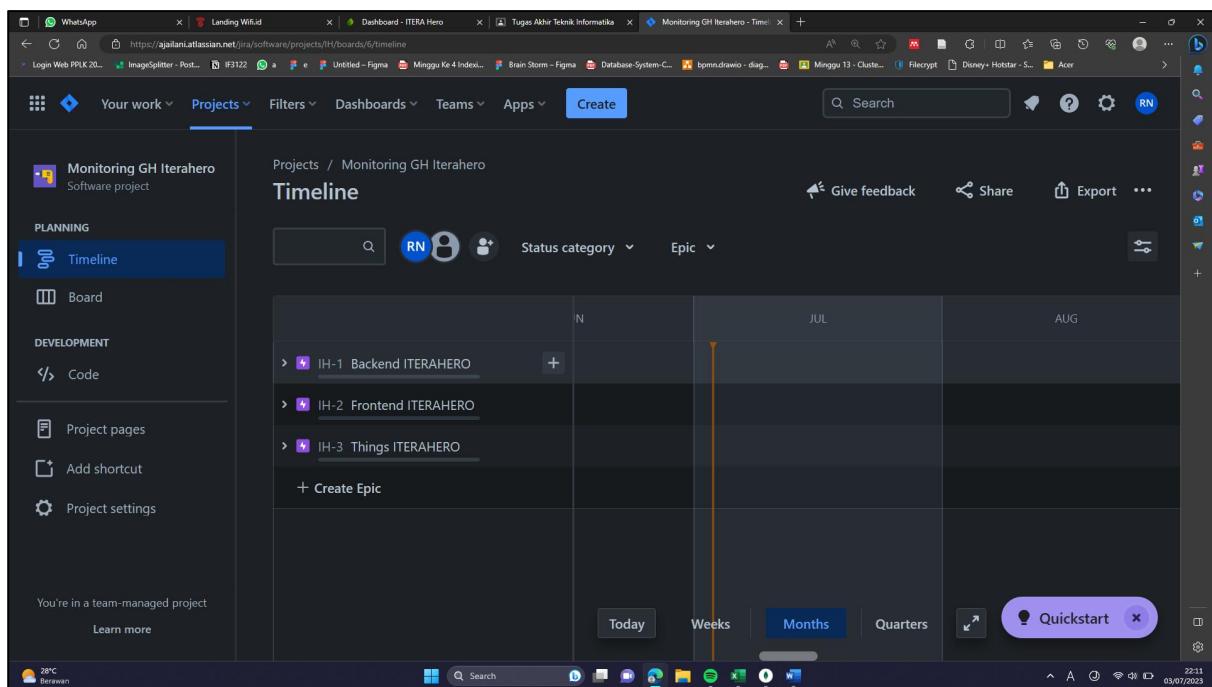
Selain menggunakan black box testing pengujian akan dilakukan menggunakan *System Usability Scale* (SUS). Target dari responden SUS ini adalah ibu Zunanik Mufidah, S.TP., M.Si dan juga orang yang terlibat dalam menjaga *greenhouse*. Jumlah Responden yang dibutukan dalam SUS adalah minimal 20 orang mahasiswa Institut Teknologi Sumatera dan juga staff dari kebun raya Institut Teknologi Sumatera.

BAB IV

HASIL DAN PEMBAHASAN

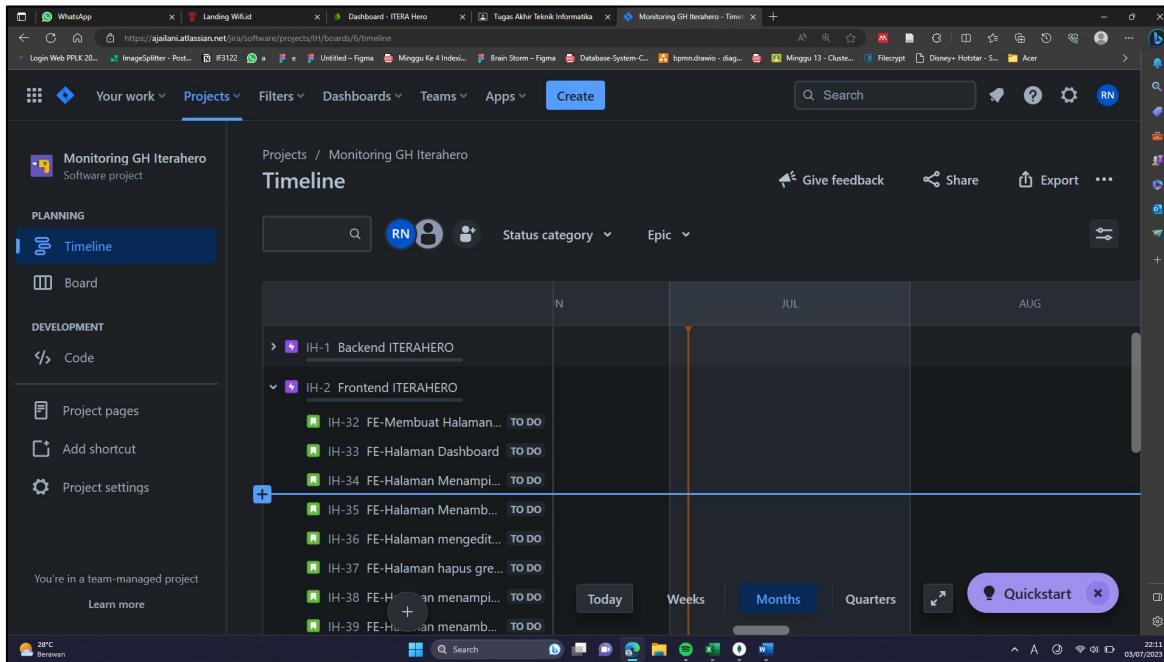
4.1 Implementasi Sistem

Implementasi sistem dengan menggunakan metode agile kanban diawali dengan menentukan *epic*. *Epic* merupakan spesifikasi umum yang akan dikerjakan, setiap *epic* akan dipecah menjadi *story*. *Story* berisi mengenai list pekerjaan yang perlu dikerjakan berdasarkan *user story* yang sebelumnya telah dibuat yakni pembuatan *Backend*, *Frontend*, dan juga *sensor* yang dapat dilihat pada gambar 4.1



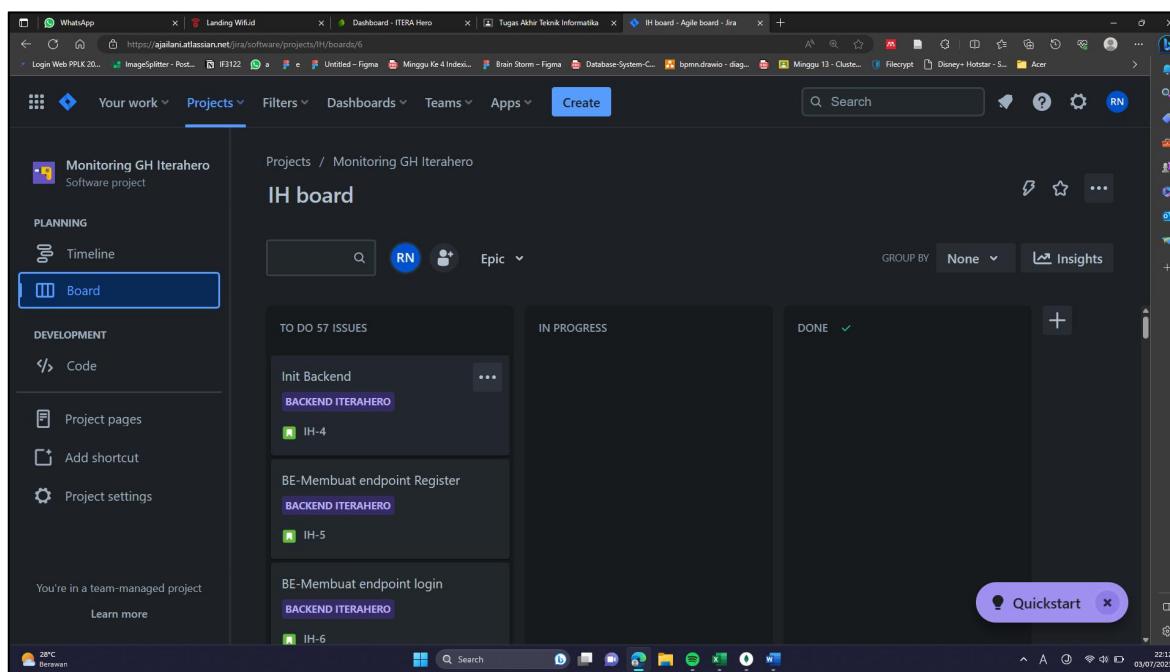
Gambar 4.1 Spesifikasi umum pembuatan sistem pada Agile Kanban

Setiap *epic* memiliki *story* yang didapatkan dari *user story* yang telah dibuat pada tabel 3.9. Setiap *story* memiliki bobot-bobot yang sesuai sehingga dapat memperkirakan waktu selesai dari *story* yang sedang dikerjakan. *Story* akan dikerjakan satu demi satu, *story* bersifat fleksibel sehingga jika terdapat fitur tambahan dapat ditambahkan *story* terbaru. *Story* dibuat berdasarkan *epic* seperti contohnya pada *epic front-end* membuat halaman login , membuat dashboard. Gambar dari *story* dapat dilihat pada gambar 4.2.



Gambar 4.2 User Story Agile Kanban

Setiap *story* akan masuk kedalam board *Kanban* pada *board* ini tim pemgembang akan mengerjakan setiap *story* yang ada pada board sehingga setiap *story* dapat dipindahkan ke sedang mengerjakan jika sedang melakukan dikerjakan pada *story* terkait dan jika telah selesai dapat dipindahkan ke *board done* sehingga pengembangan akan selesai jika semua *card* pada *board* sudah habis. Bentuk dari *board kamban* dapat dilihat pada gambar 4.3



Gambar 4.3 Board Agile Kanban

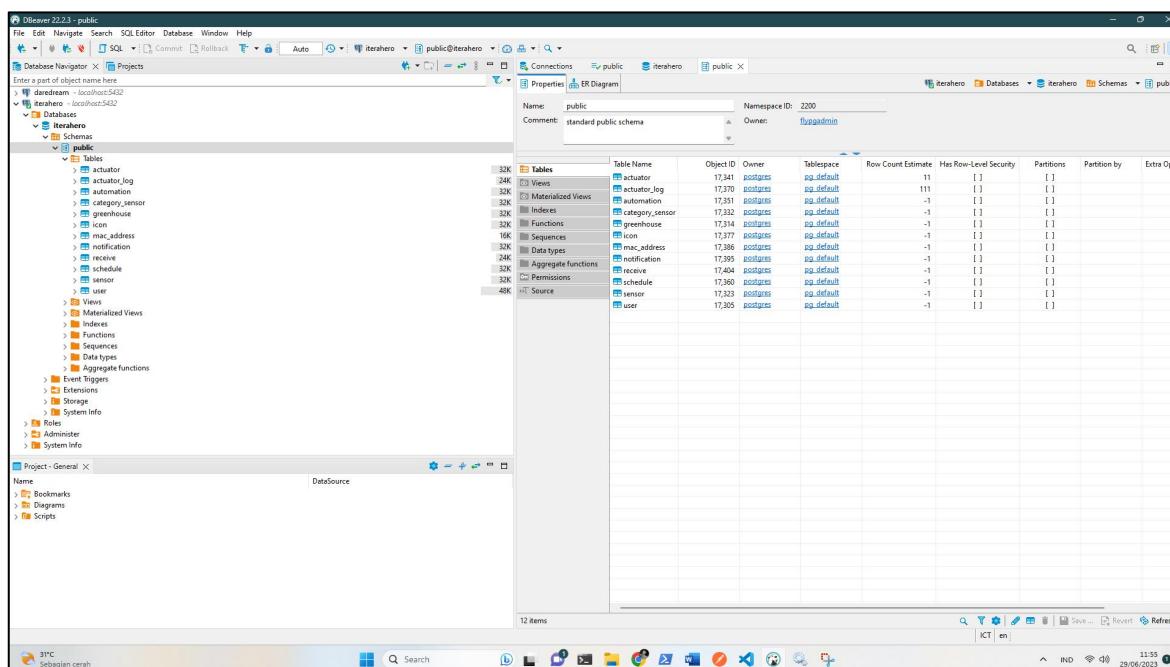
Jika Sudah Selesai maka kartu yang berisi story akan dipindahkan ke *done*, hal ini bertujuan untuk mengetahui progress dari aplikasi dan juga sebagai panduan pengembang dalam mengembangkan sistem. Untuk Poin selanjutnya adalah pembuatan dari setiap Epic yang telah dibuat menggunakan metode *Agile Kanban*.

4.1.1 Pembuatan Backend

Pembuatan backend adalah pembuatan basis data dan api untuk pertama pembuatan dilakukan dengan melakukan pembuatan pada basis data dalam penelitian ini menggunakan basis data postgresql dan juga untuk tabel pada database berdasarkan erd yang telah dibuat

4.1.1.1 Inisiasi Backend

Inisiasi yang dilakukan dalam backend adalah pembuatan basis data, basis data dibuat berdasarkan er-diagram yang telah dibuat sebelumnya, basis data digunakan untuk menyimpan data-data pengguna seperti data nama, email, username pada entitas user, selain itu menyimpan data user lainnya seperti *greenhouse* yang dimiliki hingga *sensor* yang ada pada *greenhouse*, berikut adalah hasil pembuatan basis data untuk menyimpan data dari website nantinya.



Gambar 4.4. Basis Data Sistem

Basis data berbentuk tabel-tabel dengan memiliki *field* dengan tipe data dan relasi terdapat hasil dari pembuatan basis data yang berisi mengenai tabel untuk menyimpan data. Rincian dari pembuatan basis data dapat dilihat pada tabel 4.1.

Tabel 4.1 Rincian hasil pembuatan basis data

| Nama Tabel | Field | Tipe Data | Relasi |
|-----------------|-----------------------|-----------|------------|
| Actuator | Id_actuator (serial4) | Serial4 | PK |
| | Name | text | - |
| | Status_lifecycle | Int4 | - |
| | Created_at | Timestamp | - |
| | Updated_at | Timestamp | - |
| | Icon | text | - |
| | color | text | - |
| | Automation | Int4 | - |
| | Id_greenhouse | Int4 | Greenhouse |
| | Actuator_image | Text | - |
| Actuator_log | Detail_act | Text | - |
| | positionact | Text | - |
| | Id_actuator_log | Serial4 | PK |
| | Created_at | Timestamp | - |
| Category_sensor | On_off_status | Int4 | - |
| | Id_actuator | Int4 | Actuator |
| Category_sensor | Id_category_sensor | Serial4 | PK |
| | Name | Text | - |
| Greenhouse | Id_greenhouse | Serial4 | PK |
| | Name | Text | - |

| Nama Tabel | Field | Tipe Data | Relasi |
|--------------|------------------|-----------------|--------------|
| | image | Text | - |
| | Location | Text | - |
| | Created_at | Timestamp | - |
| | Updated_at | Timestamp | - |
| | Id_user | Int4 | User |
| Icon | Id_icon | Int4 | PK |
| | name | Text | - |
| | Icon | Text | - |
| | Type | Text | - |
| | color | Text | - |
| Mac_address | Id_sensor | Int4 | Sensor |
| | Id_actuator | Int4 | Actuator |
| | Mac_address | Text | - |
| | Id_mac | Serial4 | PK |
| Notification | Id_notification | Serial4 | - |
| | Detail | Text | - |
| | Created_at | Timestamp | - |
| | Type | Text | - |
| | Status | Text | - |
| | Id_actuator | Int4 | Actuator |
| | Id_sensor | Int4 | Sensor |
| receive | Id_user | Int4 | User |
| | Id_notification | Int4 | Notification |
| | Id_receive | Serial4 | PK |
| Schedule | Id_actuator | Int4 | Actuator |
| | Start_time | Text | - |
| | Repeat | Int4 | - |
| | Id_schedule | Serial4 | PK |
| | Created_at | Created_at | - |
| | Updated_at | Updated_at | - |
| | duration | Duration | - |
| | Interval | interval | - |
| | Status_schedule | Status_schedule | - |
| | Hour | hour | - |
| | Minute | minute | - |
| Sensor | Dayofweek | Int4 | - |
| | Id_sensor | Serial4 | PK |
| | Name | Text | - |
| | Unit_measurement | Text | - |
| | Brand | Text | - |
| | Created_at | Timestamp | - |
| | Updated_at | Timestamp | - |
| | Icon | Text | - |
| | color | Text | - |
| | Id_greenhouse | Int4 | Greenhouse |

| Nama Tabel | Field | Tipe Data | Relasi |
|------------|--------------------|-----------|--------|
| | Range_min | Int4 | - |
| | Range_max | Int4 | - |
| | Id_category_sensor | Int4 | - |
| | Notify | Int4 | - |
| | Calibration | Text | - |
| | Detail | Text | - |
| | Position | Text | - |
| | Sensor_image | Text | - |
| User | Id_user | Serial4 | PK |
| | Email | Text | - |
| | name | Text | - |
| | password | Text | - |
| | username | Text | - |
| | Created_at | Timestamp | - |
| | Updated_at | timestamp | - |

Pada basis data user digunakan untuk menyimpan data dari pengguna seperti menyimpan email, *password* yang digunakan untuk *login*, lalu pada *greenhouse* digunakan agar user dapat membuat *greenhouse* dengan nama, gambar dan lokasi. Untuk setiap *greenhouse* memiliki sensor dan aktuator untuk menambahkannya perlu mengisi data-data seperti nama, satuan ukur, nilai terendah, nilai tertinggi, kalibrasi, *brand*, warna. Terdapat tabel yang digunakan untuk menyimpan kategori dari sensor untuk menampilkan sensor dalam bentuk *diagram* yang berbeda-beda dan *icon* untuk menyimpan *icon-icon* yang akan digunakan dalam sensor adanya tabel ini digunakan untuk menyimpan asset agar dapat dikelola lebih mudah seperti menghapus, menambahkan, ataupun mengedit, lalu terdapat *automation* dan juga *schudule* untuk menyimpan data otomatis. Notifikasi digunakan untuk menyimpan *notifikasi*, dan pada bagian *receive* berisi mengenai setiap user yang perlu mendapatkan notifikasi. Terdapat tabel *mac address* digunakan untuk menyimpan nilai dari mac address setiap *mikrokontroller* untuk sebagai informasi.

Selain pada postgresql sistem menggunakan database lainnya untuk menyimpan data-data sensor dan juga kondisi untuk aktuator yakni dengan membuat model dari mongodb menggunakan untuk hasil dari mongodb adalah dengan pembuatan schema, schema akan diambil dalam memasukkan data, menghapus, ataupun membaca data sehingga hasil dari pembuatan mongodb berupa schema, schema yang dibuat dibagi menjadi 2 yakni untuk sensor dan juga actuator, untuk sensor dapat dilihat pada kode

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;
```

```

const sensorSchema = new Schema(
{
  value: Number,
  id_sensor: Number,
  status: String,
},
{ timestamps: true }
);

module.exports = mongoose.model("sensor", sensorSchema);

```

Kode 4.1 Hasil pebuatan schema sensor

Pada bagian sensor sistem akan menyipakan data *id_sensor* yang digunakan untuk relasi *id_sensor* pada *basis data postgresql*, value berisi mengenai hasil pembacaan dari sensor, status yang berisi *offline* dan *online* dan juga terdapat timestamp untuk menyimpan data tanggal pembuatan data *sensor*.

Selain sensor schema sensor adapun aktuator untuk menyimpan data-data online dan offline pada actuator yang dapat dilihat pada kode

```

const mongoose = require("mongoose");
const Schema = mongoose.Schema;

const actuatorSchema = new Schema(
{
  id_actuator: Number,
  status: String,
},
{ timestamps: true }
);

module.exports = mongoose.model("actuator", actuatorSchema);

```

Kode 4.2 Hasil Pembuatan Schema Actuator

Pada bagian aktuator digunakan untuk menyimpan data-data mulai dari *id_actuator* untuk menghubungkan data dari *postgresql* dan juga data dari *mongodb* sehingga dapat menampilkan status yang sesuai akutator yang *offline* dan juga yang *online*.

Setelah berhasil membuat database Langkah selanjutnya adalah membuat API, untuk membuat API pada penelitian ini menggunakan node js dengan *framework* hapi js. Hal pertama untuk membuat hapi js adalah dengan membuat server.js

```

const Hapi = require("@hapi/hapi");

const init = async () => {
  dotenv.config();
}

const server = await Hapi.server({
  port: process.env.PORT || 8000,
  host: process.env.host || "localhost",
}

```

```

routes: {
  cors: {
    origin: ["*"],
  },
},
});

await server.register(jwt);
await mongoose.connect("mongodb://127.0.0.1:27017/iterahero", {
  useNewUrlParser: true,
});

server.auth.strategy("jwt", "jwt", {
  key: process.env.JWT_SECRET,
  expiresIn: "365d",
  validate: validate,
});

server.auth.default("jwt");

server.route(routes);

try {
  await server.start();
} catch (err) {
  console.log(err);
}

console.log(`Server is running on ${server.info.uri}`);
};

init();

```

Kode 4.3 Kode untuk Menjalankan backend

Untuk melakukan inisialisasi *backend* hal yang perlu diperhatikan adalah *framework* yang digunakan lalu setelah itu koneksi ke database, teknologi jwt, dan juga inisiasi dari *routes* lalu untuk server perlu mengetahui *host* dan juga *port* yang ditentukan terlebih dahulu agar *server backend* dapat berjalan, seperti pada kode 4.3 port yang digunakan 8000 dan juga pada host menggunakan *localhost*. Untuk *cors* digunakan untuk keamanan dari setiap endpoint agar dapat diakses oleh *frontend* tertentu atau semua. Setelah itu terdapat jwt untuk keamanan dari endpoint sehingga pengguna perlu login untuk mendapatkan endpoint yang menggunakan jwt.

4.1.1.2 Hasil Pembuatan Endpoint User

Pada bagian endpoint User berisi mengenai bagaimana sistem website untuk melakukan login agar pengguna dapat memiliki akses untuk data-datanya, data-data berisi mengenai username, email, nama, dan juga *password* dari sistem yang ada, pada *endpoint user* berisi

mengenai register untuk pengguna dapat melakukan *registrasi* pada user agar user dapat login dan mengakses aplikasi. Rincian api yang telah dibuat untuk endpoint user dapat dilihat pada tabel 4.2.

Tabel 4.2 Rincian API endpoint user

| Nama API | HTTP Method | URL | Input | Output | Keterangan |
|----------|-------------|------------------|-------------------------------|--------------------|---|
| register | POST | /api/v1/register | Username,email, name,password | Json 201 | Registrasi dari user sehingga user dapat login |
| login | POST | /api/v1/login | Email,password | accessToken, email | Api login digunakan untuk mengases data <i>greenhouse</i> |

4.1.1.3 Hasil Pembuatan Endpoint *Greenhouse*

Pada bagian endpoint *Greenhouse* digunakan untuk mengelola *greenhousei* seperti menambahkan *greenhouse*, mengedit *greenhouse*, menghapus dari *greenhouse*, dan mendapatkan data dari *greenhouse*, detail dari api ini dapat dilihat pada tabel 4.3.

Tabel 4.3 Rincian API endpoint *greenhouse*

| Nama API | HTTP Method | URL | Input | Output | Keterangan |
|------------------------------|-------------|-------------------------|----------------------|---|--|
| Upload <i>Greenhouse</i> | POST | /api/v1/greenhouse | name,image, location | Json Code 201 | Endpoint untuk menambahkan <i>greenhouse</i> |
| Get <i>Greenhouse</i> | GET | /api/v1/greenhouse | - | List dengan berisi object (Id, name, image, location,created_at) | Endpoint untuk mendapatkan list <i>greenhouse</i> berdasarkan user |
| Get <i>Greenhouse</i> Detail | GET | /api/v1/greenhouse/{id} | - | Id, name, image, location,created_at | Endpoint untuk mendapatkan detail dari <i>greenhouse</i> |

| Nama API | HTTP Method | URL | Input | Output | Keterangan |
|---------------------------|-------------|--|-----------------------|---------------|--|
| Update <i>greenhouse</i> | PUT | /api/v1/ <i>greenhouse</i> / <i>{id}</i> | Name,im age,locati on | Json Code 200 | Enpoint untuk mengedit <i>greenhouse</i> |
| Delete <i>Greenhou se</i> | DELETE | /api/v1/ <i>greenhouse</i> / <i>{id}</i> | - | Json Code 200 | Endpoint untuk menghapus <i>greenhouse</i> |

4.1.1.4 Hasil Pembuatan Endpoint Sensor

Pada bagian endpoint *Greenhouse* digunakan untuk mengelola sensor seperti menambahkan *endpoint* dari menambahkan, mengedit, menghapus, mendapatkan data dari sensor, data sensor berdasarkan *greenhouse*, detail dari api ini dapat dilihat pada tabel 4.4.

Tabel 4.4 Rincian API Endpoint Sensor

| Nama API | HTTP Method | URL | Input | Output | Keterangan |
|---------------|-------------|----------------|---|---------------|---|
| Upload Sensor | POST | /api/v1/sensor | Name, unit_measurement, brand, color, id_greenhouse, range_min, range_max, id_category_sensor, icon, detail, position, sensor_image | Json code 201 | Endpoint untuk menambahkan sensor berdasarkan <i>greenhouse</i> |

| Nama API | HTTP Method | URL | Input | Output | Keterangan |
|---------------------------------------|-------------|------------------------|--|---|---|
| Get Sensor By Id <i>greenhouse</i> | GET | /api/v1/sensor | - | List Json berisi object (Name, unit_measurement, brand, color, id_greenhouse, range_min, range_max, category, icon, detail, position, sensor_image, created_at, updated_at) | Endpoint untuk mengambil list sensor dalam bentuk json berdasarkan id <i>greenhouse</i> |
| Get Sensor Detail | GET | /api/v1/sensor/{id} | - | Name, unit_measurement, brand, color, id_greenhouse, range_min, range_max, id_category_sensor, icon, detail, position, sensor_image, created_at, updated_at | Endpoint untuk mendapatkan detail dari sensor |
| Update Sensor | PUT | /api/v1/sensor/{id} | Name, unit_measurement, brand, color, range_min, range_max, id_category_sensor, icon, detail, position, sensor_image | Json code 200 | Endpoint untuk mengedit sensor berdasarkan id sensor |
| Delete Sensor | DELETE | /api/v1/sensor/{id} | - | Json code 200 | Endpoint untuk menghapus sensor |
| Upload Category Sensor | POST | api/v1/category/sensor | name | Json code 201 | Endpoint untuk mengupload category tampilan dari sensor |

| Nama API | HTTP Method | URL | Input | Output | Keterangan |
|---------------------|-------------|------------------------|-------|-----------------------------------|--|
| Get Category Sensor | GET | api/v1/category/sensor | - | List Json berisi object (Name,id) | Endpoint untuk mendapatkan category dari tampilan sensor |

4.1.1.5 Hasil Pembuatan Endpoint Actuator

Pada bagian endpoint aktuator digunakan untuk mengelola aktuator mulai dari menhapus, mengedit, menambahkan, dan mendapatkan data dari aktuator, dikarenakan relasi aktuator berdasarkan *greenhouse* sehingga beberapa endpoint didapatkan berdasarkan *greenhouse* sehingga perlu adanya *greenhouse* terlebih dahulu, detail dari api ini dapat dilihat pada tabel 4.5.

Tabel 4.5 Rincian API endpoint actuator

| Nama API | HTTP Method | URL | Input | Output | Keterangan |
|-----------------------------------|-------------|----------------------|---|---|---|
| Upload Actuator | POST | api/v1/actuator | Name ,color, id_greenhouse , icon, detailact, positionact, actuator_image | Json code 201 | Endpoint untuk menambahkan aktuator pada sebuah <i>greenhouse</i> |
| Get Actuator by <i>Greenhouse</i> | GET | api/v1/actuator | - | Name ,color, id_greenhouse, icon, detailact, positionact, actuator_image , created_at, Updated_at | Endpoint untuk mendapatkan aktuator berdasarkan <i>greenhouse</i> |
| Get Actuator Detail | GET | api/v1/actuator/{id} | - | Name ,color, id_greenhouse, icon, detailact, positionact, actuator_image , created_at, Updated_at | Endpoint untuk mendapatkan detail dari sebuah aktuator |

| Nama API | HTTP Method | URL | Input | Output | Keterangan |
|-----------------|-------------|----------------------|---|---------------|---|
| Edit Actuator | GET | api/v1/actuator/{id} | Name ,color, icon, detailact, positionact, actuator_image | Json code 200 | Endpoint untuk mengedit sebuah aktuator |
| Delete Actuator | DELETE | api/v1/actuator/ | - | Json code 200 | Endpoint untuk menghapus aktuator |

4.1.1.6 Hasil Pembuatan Endpoint Notifikasi

Pada bagian endpoint notifikasi digunakan untuk mengelola notifikasi, seperti menambahkan notifikasi jika sensor ataupun aktuator ingin dilakukan notifikasi dengan *endpoint* ini, lalu terdapat hapus, dan juga mendapatkan data dari notifikasi digunakan agar user dapat melihat hasil dari notifikasi yang telah diunggah sensor ataupun aktuator detail dari api ini dapat dilihat pada tabel 4.6.

Tabel 4.6 Rincian API endpoint notifikasi

| Nama API | HTTP Method | URL | Input | Output | Keterangan |
|-----------------------------|-------------|--------------------------|--------------------|--|---|
| Upload Notification | POST | api/v1/notification | Detail,type,status | Json code 201 | Endpoint untuk mengupload sebuah notifikasi |
| Get Notification By User Id | GET | api/v1/ notification | - | Id, detail,type, created_at, status, id Sensor | Endpoint untuk mengambil list notifikasi berdasarkan user |
| Get Notification Detail | GET | api/v1/notification/{id} | - | Id, detail,type, created_at, status, id | Endpoint untuk mengambil detail dari notifikasi |

| Nama API | HTTP Method | URL | Input | Output | Keterangan |
|---------------------|-------------|--------------------------|-------|---------------|-------------------------------------|
| | | | | Sensor | |
| Delete Notification | DELETE | api/v1/notification/{id} | - | Json code 200 | Endpoint untuk menghapus notifikasi |

4.1.1.7 Hasil Pembuatan Endpoint Icon

Pada bagian endpoint *Icon* digunakan untuk mengelola icon seperi menambahkan data-data dari *icon* yang digunakan untuk tampilan dari sensor dan aktuator agar memiliki icon terdapat 2 endpoint yakni menambahkan *icon* dan juga untuk mendapatkan *icon*, detail dari api ini dapat dilihat pada tabel 4.7.

Tabel 4.7 Rincian API endpoint icon

| Nama API | HTTP Method | URL | Input | Output | Keterangan |
|-------------|-------------|-------------|-------------------------|-------------------------|--|
| Upload Icon | POST | api/v1/icon | Name, icon, type, color | Json code 201 | Endpoint untuk menambahkan icon pada sensor ataupun aktuator |
| Get Icon | GET | api/v1/icon | - | Name, icon, type, color | Endpoint untuk mendapatkan icon |

4.1.1.8 Hasil Pembuatan Endpoint Automation

Pada bagian endpoint Automation digunakan untuk mengelola automation yang digunakan untuk menyalakan *aktuator* berdasarkan sensor *endpoint* ini berisi mengenai menambahkan, mengambil, mengedit, dan juga menghapus dari *automation*. Endpoint ini dapat mengatur otomatis berdasarkan sensor, detail dari api ini dapat dilihat pada tabel 4.8,

Tabel 4.8 Rincian API endpoint automation

| Nama API | HTTP Method | URL | Input | Output | Keterangan |
|-------------------|-------------|-------------------|--|---------------|--|
| Upload automation | POST | api/v1/automation | Id_actuator, id_sensor, condition, status_lifecycle, constants | Json code 201 | Endpoint untuk menambahkan otomatis berdasarkan sensor |

| Nama API | HTTP Method | URL | Input | Output | Keterangan |
|-----------------------|-------------|------------------------|--|--|---|
| Get Automation | GET | api/v1/automation | - | Data : [{Sensor{id_sensor, name}, actuator {id_actuator, name}, condition, status_lifecycle, created_at, constanta, id_automation}] | Endpoint untuk mendapatkan list otomatis berdasarkan sensor |
| Get Detail Automation | GET | api/v1/automation/{id} | - | Sensor{}, actuator {}, condition, status_lifecycle, created_at, constanta, id_automation | Endpoint untuk mendapatkan detail dari automation |
| Edit Automation | PUT | api/v1/automation/{id} | Id_actuator, id_sensor, condition, status_lifecycle, constanta | Json code 200 | Endpoint untuk mengedit autmatis berdasarkan sensor |
| Delete Automation | DELETE | api/v1/automation/{id} | - | Json code 200 | Endpoint untuk menghapus otomatis berdasarkan sensor |

4.1.1.9 Hasil Pembuatan Endpoint Scheduling

Pada bagian endpoint *scheduling* digunakan untuk mengelola aktuator berdasarkan penjadwalan mulai dari menambahkan jadwal, mendapatkan jadwal, mengedit jadwal dan terakhir adalah menghapus jadwal, jadwal ditambahkan oleh *user* lalu ditambahkan ke *basis data* melalui *endpoint*, mendapatkan data digunakan untuk menampilkan data pada *frontend* berdasarkan jadwal yang sudah ditambahkan, selain itu mendapatkan dilakukan pada *backend* sehingga jika waktu sudah tepat dengan jadwal akan melakukan aksi, detail dari api ini dapat dilihat pada tabel 4.9.

Tabel 4.9 Rincian API endpoint scheduling

| Nama API | HTTP Method | URL | Input | Output | Keterangan |
|-----------------------|-------------|----------------------|---|---|--|
| Upload scheduling | POST | api/v1/schedule | Start,inteval,repeat,duration,id_actuator | Json code 201 | Endpoint untuk mengupload penjadwalan berdasarkan waktu |
| Get All Scheduling | GET | api/v1/schedule | - | Start,inteval,repeat,duration,actuator{id,name,automation_status}.hour,minute,dayOfWeek | Endpoint untuk mengambil seluruh data penjadwalan berdasarkan waktu untuk aktuator |
| Get Detail Scheduling | GET | api/v1/schedule/{id} | - | Start,inteval,repeat,duration,actuator{id,name,automation_status}.hour,minute,dayOfWeek | Endpoint untuk mendapatkan detail dari penjadwalan |
| Edit Scheduling | PUT | api/v1/schedule/{id} | - | Json code 200 | Endpoint untuk mengedit data otomatis berdasarkan waktu(penjadwalan) |
| Delete Scheduling | DELETE | api/v1/schedule/{id} | - | Json code 200 | Endpoint untuk menghapus data penjadwalan |

4.1.1.10 Hasil Pembuatan Endpoint Actuator Log

Pada bagian endpoint Aktuator log digunakan untuk mengelola aktuator log, pada endpoint ini mengelola dilakukan mulai dari menambahkan, mengedit, mengupdate, dan juga

mendapatkan data *aktuator log*, aktuator log digunakan untuk menyimpan data aktuator yang menyala dan juga mati sehingga dapat dilihat di *front end* data-data yang ada seperti jika aktuator nyala ataupun mati, detail dari api ini dapat dilihat pada tabel 4.10.

Tabel 4.10 Rincinan API endpoint *actuator-log*

| Nama API | HTTP Method | URL | Input | Output | Keterangan |
|---------------------------------|-------------|---------------------|-------------------------------|--|---|
| Upload Actuator Log | POST | api/v1/actuator-log | Id_actuator, on_off_status | Json code 201 | Endpoint untuk menambahkan data menyala ataupun mati dari aktuator |
| Get Actuator Log By ID Actuator | GET | api/v1/actuator-log | - | Id_actuator, on_off_status, created_at | Endpoint untuk mendapatkan list aktuator yang menyala dan mati berdasarkan id aktuator. |
| Actuator Log Detail | GET | api/v1/actuator-log | - | Id_actuator, on_off_status, created_at | Endpoint untuk mendapatkan detail dari aktuator yang menyala dan mati |
| Actuator Logs Today | GET | api/v1/actuator-log | - | Id_actuator, on_off_status, created_at | Endpoint untuk mendapatkan jumlah data aktuator menyala dan mati pada hari ini |

4.1.1.11 Hasil Pembuatan Endpoint Sensor Log

Pada bagian endpoint *Sensor Log* digunakan untuk menyimpan data-data dari *sensor* yang didapatkan berdasarkan *mikrokontroller* yang nantinya dapat ditampilkan dalam bentuk

grafik dan juga selain itu untuk mendapatkan data dari sensor yang dapat langsung dilihat pada bagian *frontend* secara *realtime*, detail dari api ini dapat dilihat pada tabel 4.11.

Tabel 4.11 Rincian API endpoint sensor-log

| Nama API | HTTP Method | URL | Input | Output | Keterangan |
|------------------------|-------------|----------------------|-------|---|--|
| Get Sensor from broker | GET | api/v1/sensor_broker | - | _id, value, id_sensor, status, createdAt, updatedAt | Endpoint untuk mendapatkan nilai sensor terbaru berdasarkan alat |

4.1.1.12 Hasil Pembuatan Endpoint Dashboard

Pada bagian endpoint *dashboard* berisi mengenai data-data yang akan ditampilkan pada dashboard mulai dari jumlah *greenhouse*, jumlah *sensor*, dan juga jumlah dari actuator yang ada, untuk detail dari api ini dapat dilihat pada tabel 4.12.

Tabel 4.12 Rincian endpoint dashboard

| Nama API | HTTP Method | URL | Input | Output | Output |
|---------------------|-------------|--------------------|-------|------------------------------------|--|
| Get Count Dashboard | GET | api/v1/dashhhboard | - | Greenhouse, sensor, actuator, name | Endpoint untuk mendapatkan data dari dashboard mulai dari jumlah <i>greenhouse</i> , jumlah <i>sensor</i> , dan juga nama user |

4.1.1.13 Hasil Pembuatan Endpoint Grafik

Pada bagian endpoint grafik digunakan untuk mendapatkan data dari sensor yang digunakan untuk mendapatkan nilai serta label yang digunakan untuk membuat grafik pada bagian *frontend* terdapat query pada endpoint grafik dengan menggunakan tanda tanya untuk mendapatkan data berdasarkan harian, mingguan, dan juga tahunan, untuk detail dari api ini dapat dilihat pada tabel 4.13.

Tabel 4.13 Rincian API endpoint grafik

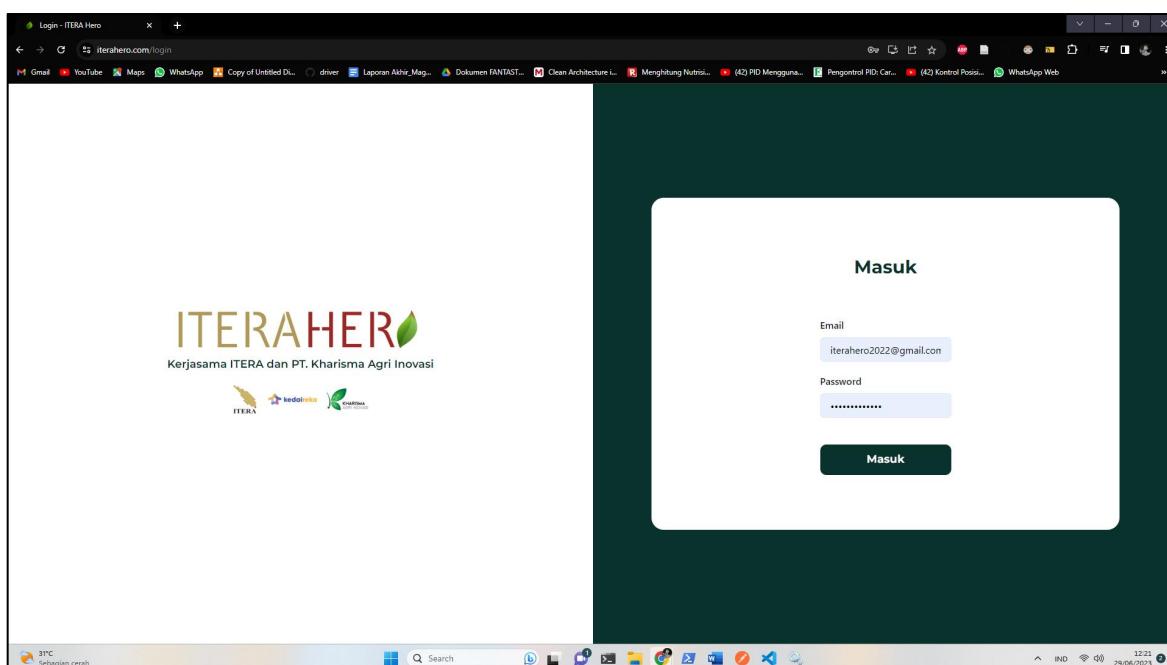
| Nama API | HTTP Method | URL | Input | Output | Output |
|-------------------|-------------|------------------|-------|------------|--|
| Get Grafik Sensor | GET | api/v1/grafik/18 | - | Label,data | Endpoint untuk mendapatkan data untuk membuat grafik berisi label dan juga data. |

4.1.2 Implementasi Front End

Frontend merupakan tampilan yang sebelumnya di desain menjadi kode agar dapat diakses oleh semua pengguna, pada sub bab ini berisi mengenai implementasi tampilan yang telah dibuat beserta pengujian berikut adalah rincian dari semua tampilan dan pengujian dari sistem *monitoring greenhouse*.

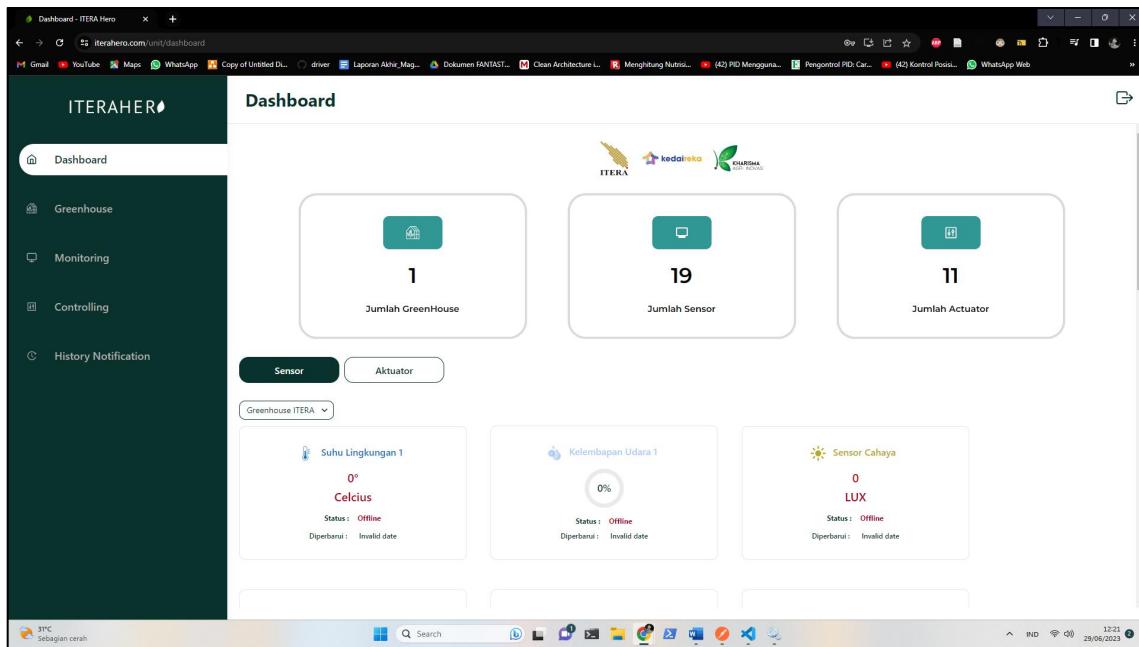
4.1.2.1 Implementasi Halaman Login

Halaman Login berisi mengenai email dan juga password yang telah diregistrasi, registrasi bersifat tertutup sehingga tidak semua user dapat melakukan registrasi karena jika melakukan register yang baru perlu bantuan dari developer untuk menambahkan alat dan juga actuator, setelah mengisi email dan password yang benar tekan tombol submit seperti pada gambar 4.5.



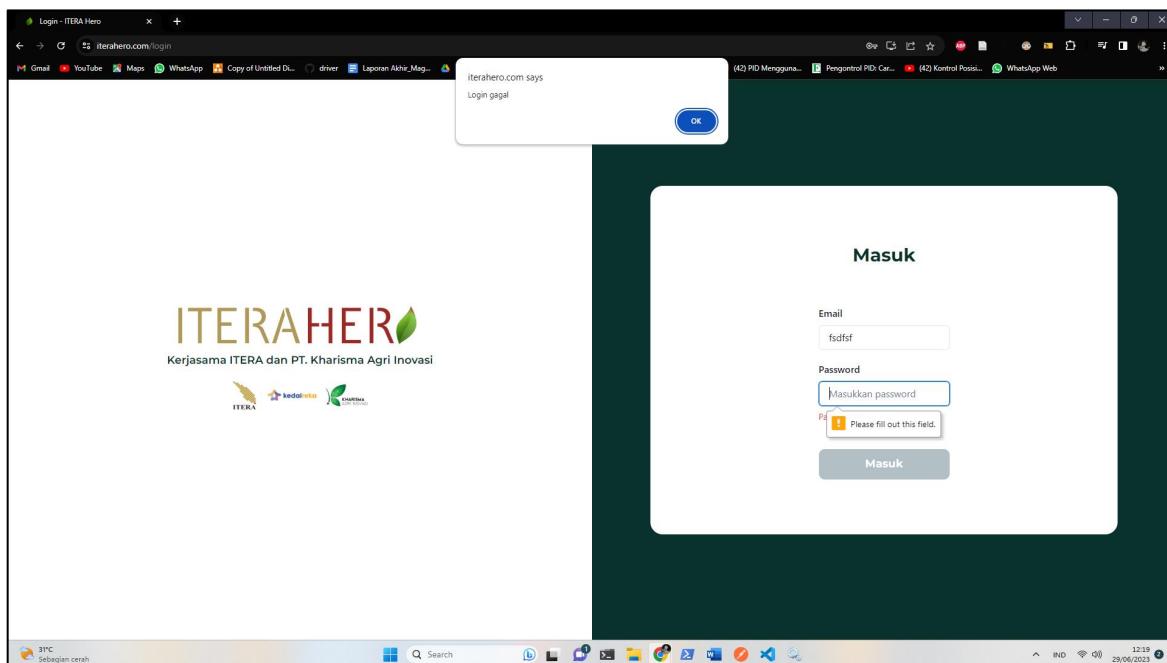
Gambar 4.5. Tampilan Front End halaman Login

Jika sistem menerima email dan password yang benar maka akan masuk ke dalam halaman dashboard halaman ini berisi mengenai jumlah aktuator, jumlah sensor, dan juga jumlah dari *greenhouse*. Informasi yang ada pada *dashboard* digunakan untuk melakukan *monitoring* dan juga control dari aktuator yang ada seperti seperti pada gambar 4.6.



Gambar 4.6 Tampilan Front end Dashboard

Akan tetapi jika sistem yang dimasukkan salah maka login akan gagal dan menerima hasil login gagal pada bagian alert hal ini terjadi karena email ataupun password salah dimasukkan seperti pada gambar 4.7.



Gambar 4.7 Tampilan Front End Jika Login gagal

Berdasarkan fitur yang telah dibuat system berhasil melakukan login dan juga jika salah maka gagal, sehingga dapat disimpulkan pengujian yang dilakukan telah berhasil, rincian pengujian pada halaman login dapat dilihat pada tabel 4.14.

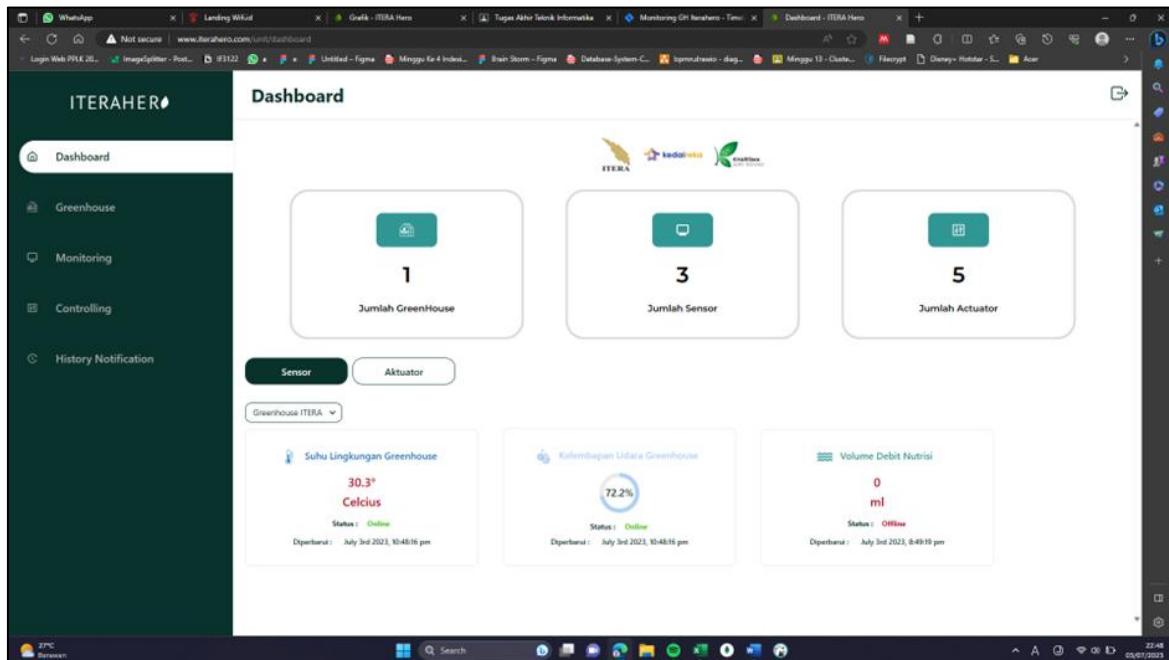
Tabel 4.14. Pegujian Black box pada halaman login

| Fitur | Case | Indikator Keberhasilan | Hasil |
|-------|--|---|----------|
| Login | Menginput sesuai dengan email dan password | Masuk ke halaman home | Berhasil |
| | Menginput tidak sesuai dengan email dan password | Kembali ke login dengan pemberitahuan login gagal | Berhasil |

4.1.2.2 Implementasi Halaman Dashboard

Pada halaman dashboard digunakan untuk menampilkan jumlah sensor dan actuator serta *greenhouse* yang dimiliki oleh user yang telah berhasil login, dashboard digunakan untuk melakukan monitoring dan juga melakukan kendali pada greenhouse berdasarkan sensor dan juga *actuator* yang dimiliki dari dashboard, pada halaman ini dibuat untuk memudahkan

pengguna mengetahui informasi mengenai IoT yang ada pada *greenhouse*. Implementasi dari halaman front end ini dapat dilihat pada gambar 4.8.



Gambar 4.8 Tampilan Front End halaman Dashboard

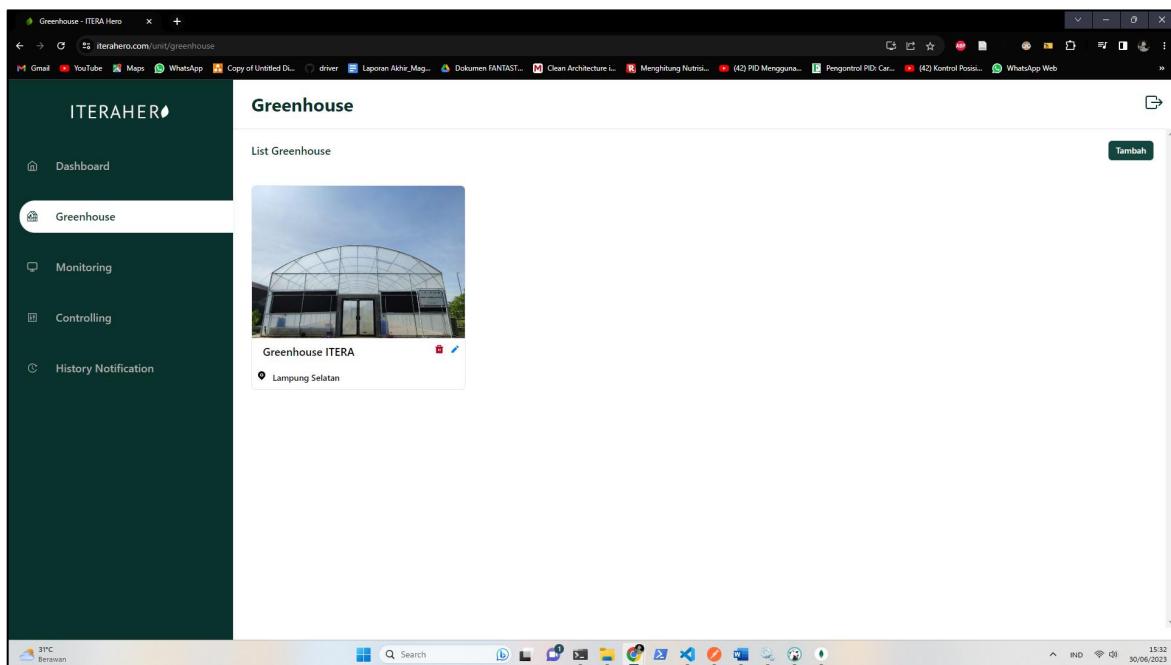
Berdasarkan gambar 4.8. sistem berhasil menampilkan jumlah sensor, actuator dan juga *greenhouse*, sehingga system berhasil pada pengujian *blackbox*. Rincian pengujian *blackbox* dapat dilihat pada tabel 4.15.

Tabel 4.15 Pengujian black box pada halaman Dashboard

| Fitur | Case | Indikator Keberhasilan | Hasil |
|-----------|----------------------------|---|----------|
| Dashboard | Masuk ke halaman dashboard | Menampilkan jumlah sensor, actuator dan <i>greenhouse</i> | Berhasil |

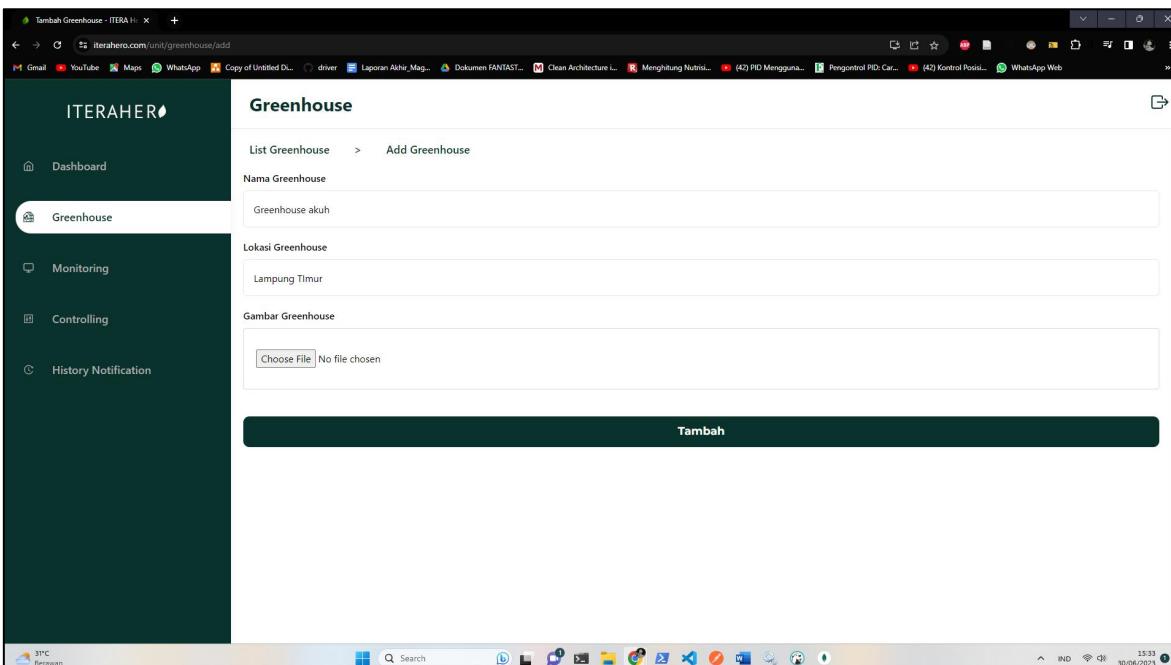
4.1.2.3 Implementasi Halaman *Greenhouse*

Halaman *greenhouse* akan menampilkan seluruh *greenhouse* yang dimiliki oleh user, halaman ini digunakan untuk mengelola *greenhouse* detail dari tampilan *greenhouse* yang dimiliki user berupa gambar, nama dan juga lokasi, user dapat memiliki lebih dari satu *greenhouse* sehingga user dapat menambahkan *greenhouse* sebanyak-banyaknya, gambar dari halaman *greenhouse* dapat dilihat pada gambar 4.9.



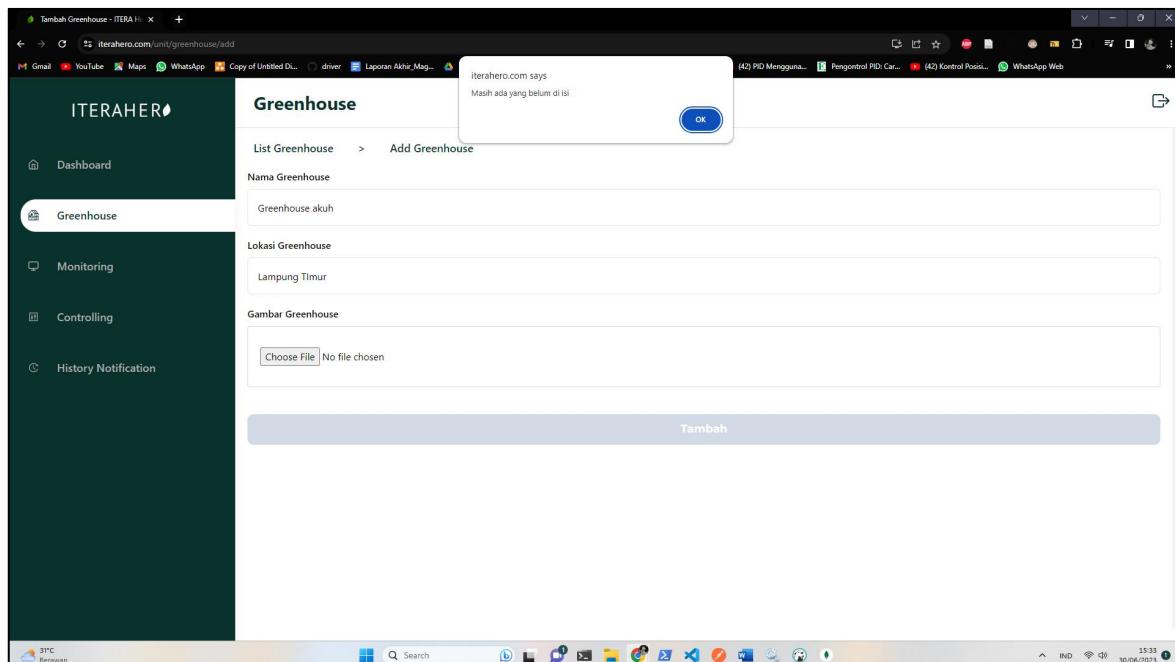
Gambar 4.9 Tampilan Front end halaman monitoring

Pada halaman *greenhouse* terdapat tanda tambah jika di klik maka akan masuk kehalaman menambah *greenhouse* untuk Menambah *Greenhouse* perlu mengisikan nama, lokasi dan juga gambar dari *greenhouse*, detail dari form penambahan *greenhouse* dapat dilihat pada gambar 4.10.



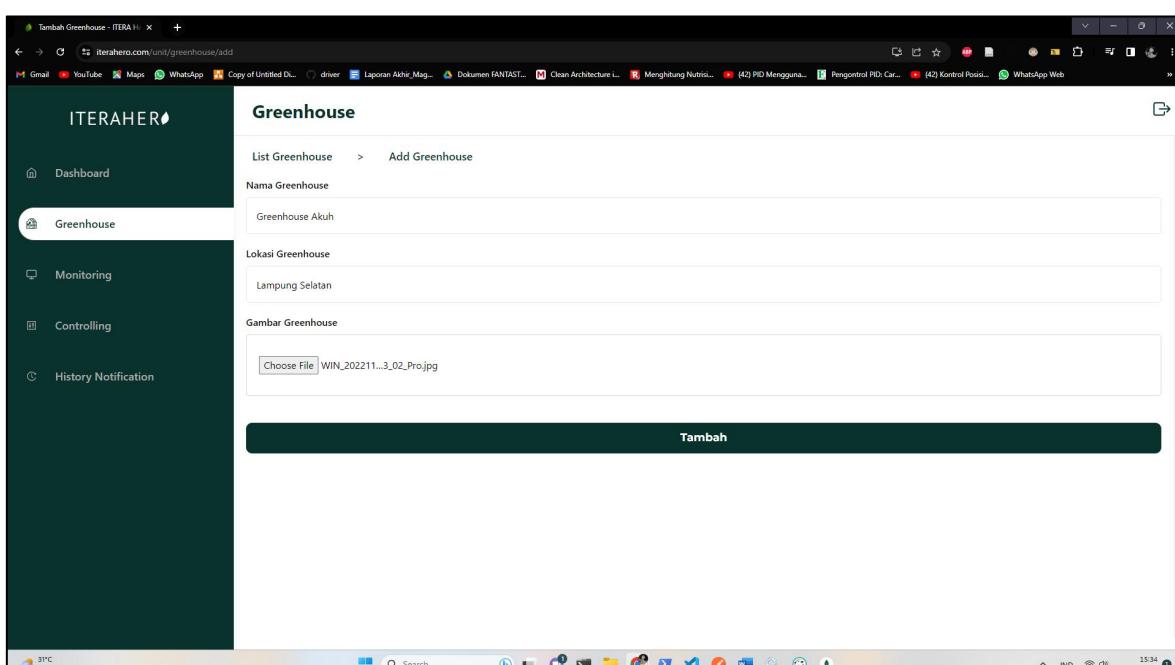
Gambar 4.10 Tampilan Front End halaman menambah *greenhouse*

Pada saat menambahkan *greenhouse* perlu mengisi seluruh field yang ada pada *greenhouse* jika tidak akan muncul pemberitahuan bahwa ada field yang kosong yakni field gambar yang belum diisi seperti pada gambar 4.11.



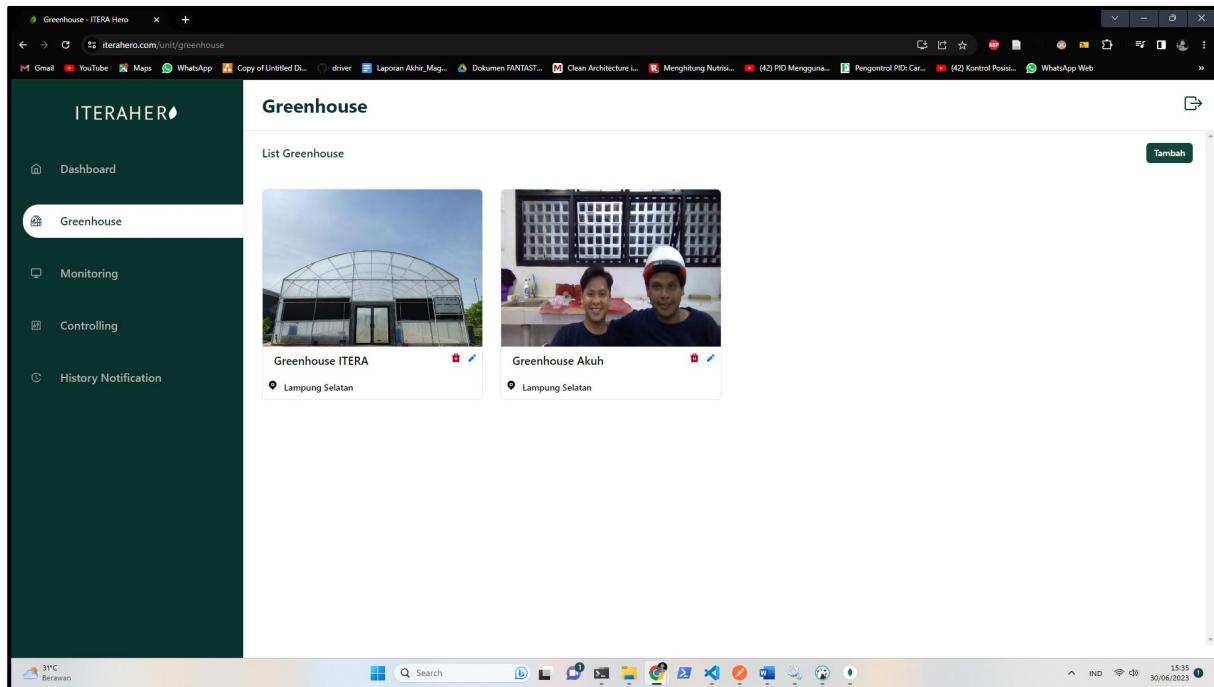
Gambar 4.11 Tampilan jika tidak mengisi data pada tambah *greenhouse*

Agar dapat menambah *greenhouse* perlu mengisi seluruh *field* yang ada pada sistem mulai dari nama, lokasi dan juga gambar yang dapat dilihat pada gambar 4.12.



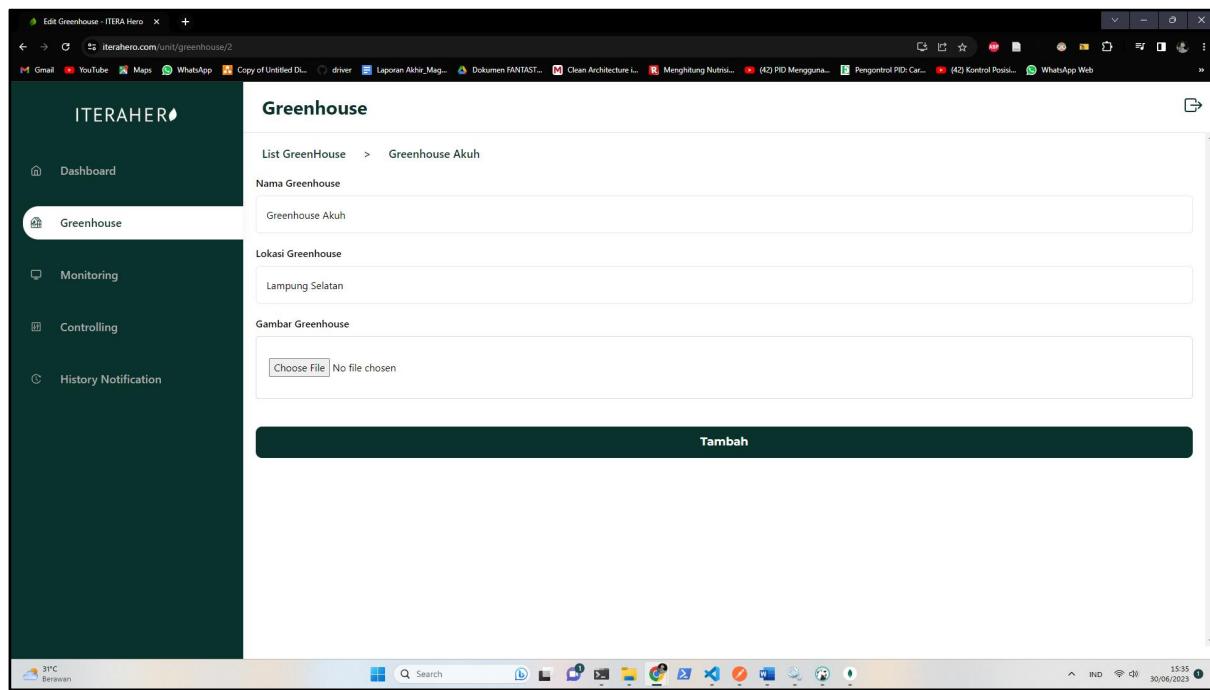
Gambar 4.12 Mengisi seluruh field menambah *greenhouse*

Setelah mengisi dengan menekan tombol tambah yang ada pada website akan menambahkan *greenhouse* pada sistem sehingga *greenhouse* berhasil ditambah maka akan tampil pada halaman utama dari *greenhouse* seperti pada gambar 4.13.



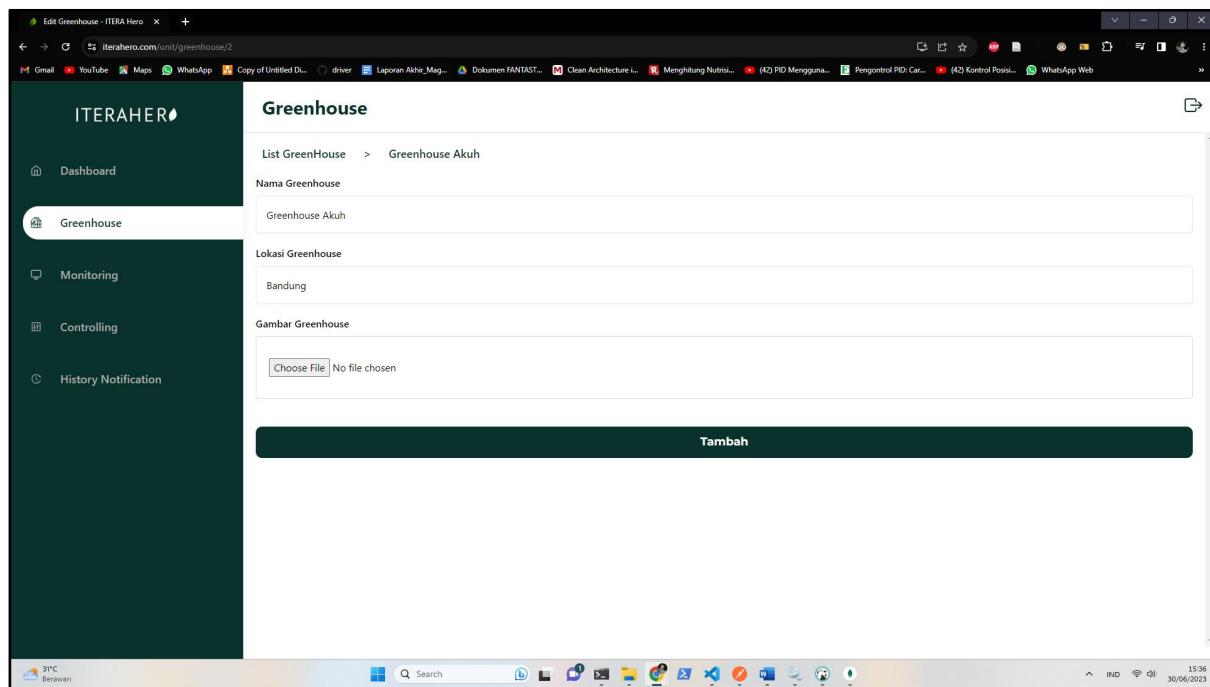
Gambar 4.13. Tampilan jika berhasil menambahkan *greenhouse*

Lalu jika mengedit maka akan tampil data dari *greenhouse* yang telah diisi sebelumnya sehingga edit dapat diubah sebagian saja seperti pada gambar 4.14.



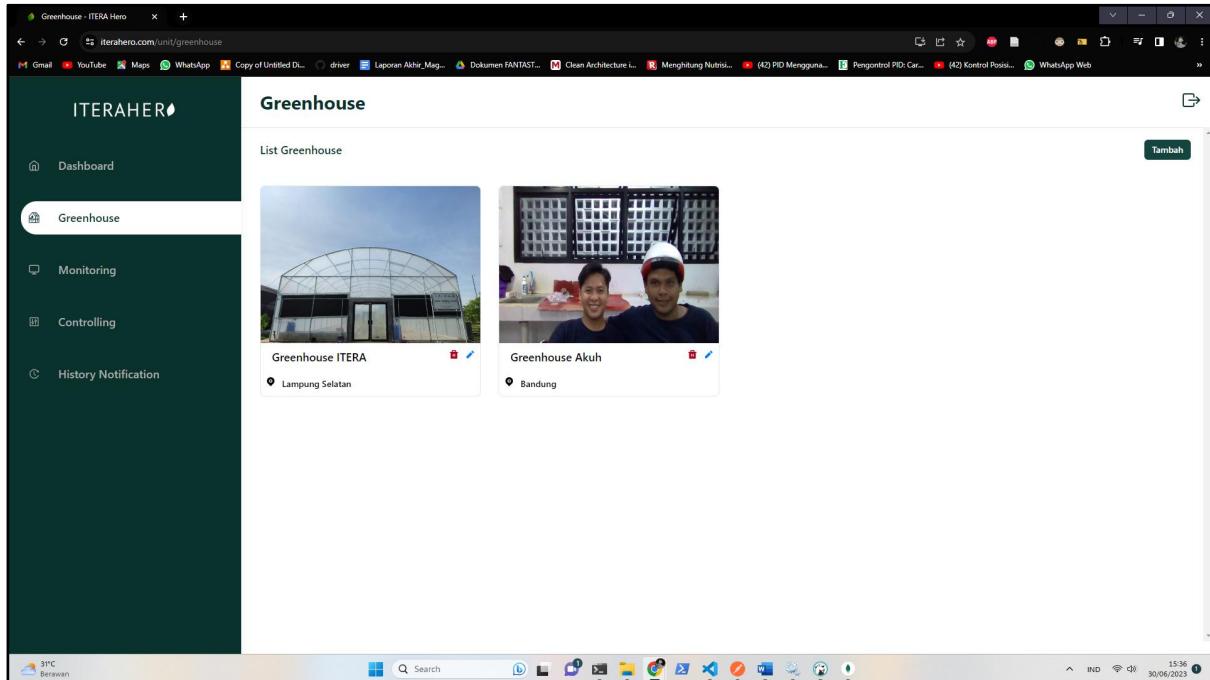
Gambar 4.14 Tampilan Front End mengedit *greenhouse*

Lalu jika mengedit maka akan tampil data dari *greenhouse* yang telah diisi sebelumnya sehingga edit dapat dirubah sebagian saja seperti merubah lokasi jadi bandung seperti pada gambar 4.15.



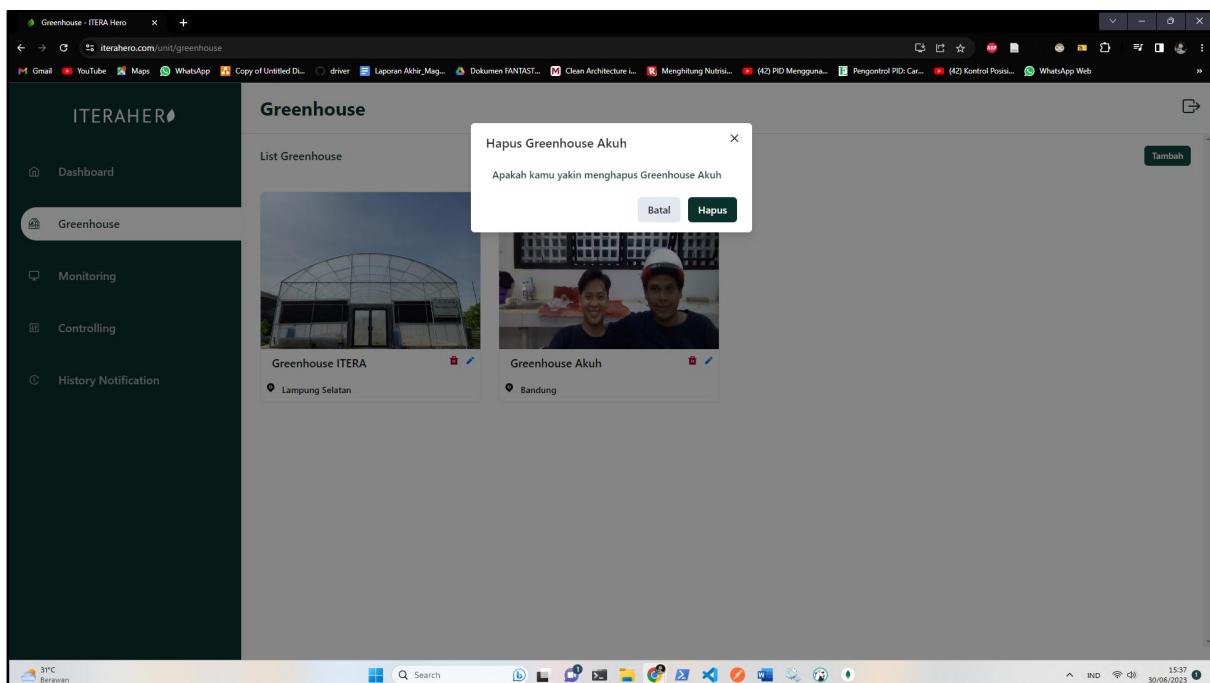
Gambar 4.15. Tampilan Mengubah field lokasi *greenhouse*

Hasil dari *edit* akan merubah lokasi dari *greenhouse* yang ada pada halaman utama seperti pada gambar 4.16.



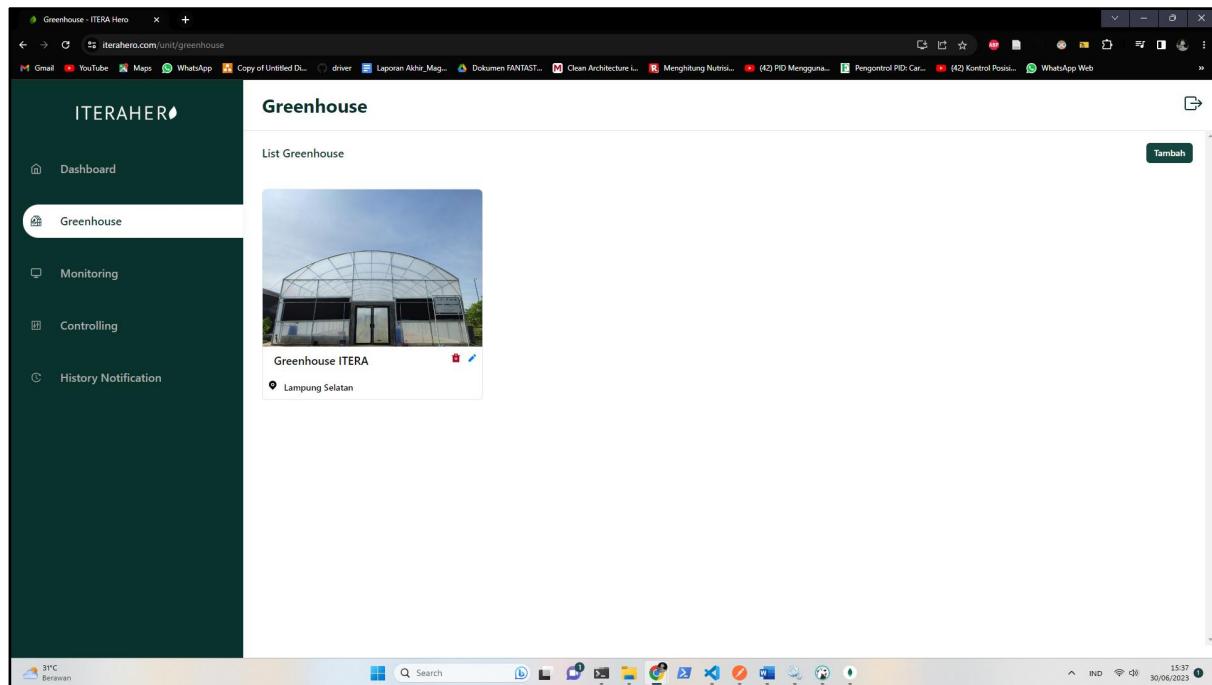
Gambar 4.16. Hasil mengedit lokasi *greenhouse*

Terakhir jika menghapus *greenhouse* maka akan tampil halaman *alert* jika dibatalkan maka tidak dihapus, dan juga jika hapus akan terhapus seperti pada gambar 4.17.



Gambar 4.17 Peringatan saat menghapus *greenhouse*

Setelah ditekan tombol hapus maka system akan berhasil menghapus *greenhouse* yang ingin dihapus maka *greenhouse* yang telah dihapus akan hilang dari basis data dan tidak akan muncul pada halaman *greenhouse* seperti pada gambar 4.18.



Gambar 4.18 Halaman *greenhouse* setelah *greenhouse* di hapus

Lalu berdasarkan pengujian yang dilakukan *greenhouse* dapat dikelola dengan baik mulai dari menampilkan, menambah , mengedit dan menghapus telah berhasil. Rincian dari pengujian dapat dilihat pada tabel 4.16.

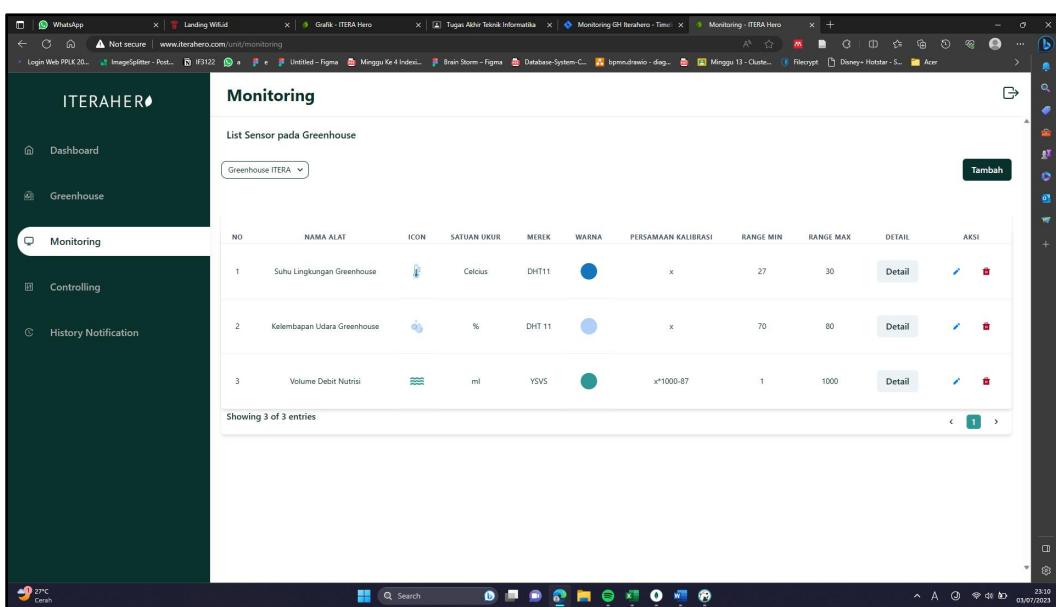
Tabel 4.16 Rincian pengujian black box pada halaman *greenhouse*

| Fitur | Case | Indikator Keberhasilan | Hasil |
|-------------------------------|--|-------------------------------------|----------|
| Menampilkan <i>greenhouse</i> | ke halaman <i>greenhouse</i> dan menampilkan <i>greenhouse</i> | <i>Greenhouse</i> tampil | Berhasil |
| Menambah <i>greenhouse</i> | Menginput <i>greenhouse</i> dengan data lengkap | Berhasil menambah <i>greenhouse</i> | Berhasil |
| | Menginput <i>greenhouse</i> dengan data yang tidak lengkap | tidak dapat di submit | Berhasil |
| Delete <i>greenhouse</i> | Menghapus <i>Greenhouse</i> | <i>Greenhouse</i> terhapus | Berhasil |

| Fitur | Case | Indikator Keberhasilan | Hasil |
|------------------------|----------------------------|---------------------------|----------|
| Edit <i>greenhouse</i> | Mengedit <i>greenhouse</i> | <i>Greenhouse</i> teredit | Berhasil |

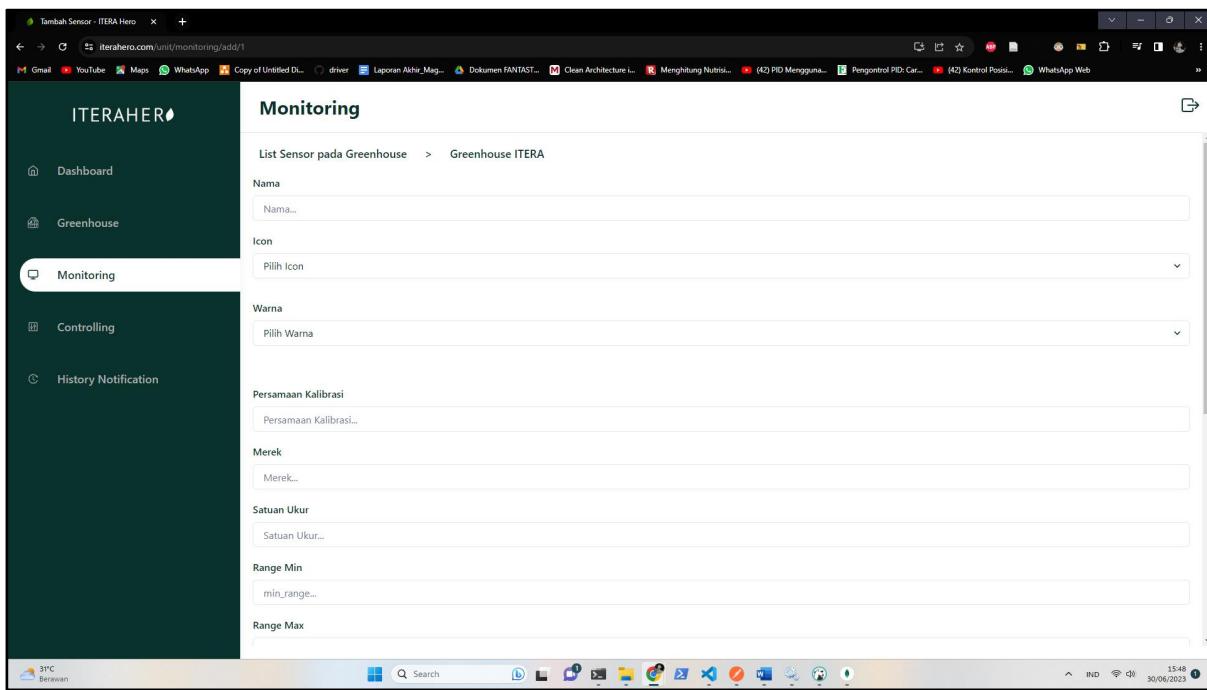
4.1.2.4 Implementasi Halaman Sensor

Halaman monitoring pada website digunakan untuk mengelola list dari sensor berdasarkan *greenhouse* mulai dari menampilkan data sensor mulai dari nama, icon, merek dari sensor, satuan ukur dari sensor, batas atas yang digunakan untuk set kondisi ideal, dan juga batas bawah dari kondisi ideal, lalu detail dari *greenhouse*, dan aksi mulai dari menghapus dan mengedit *sensor* berdasarkan *greenhouse*, tampilan dari halaman monitoring yang dapat dilihat pada gambar 4.19.



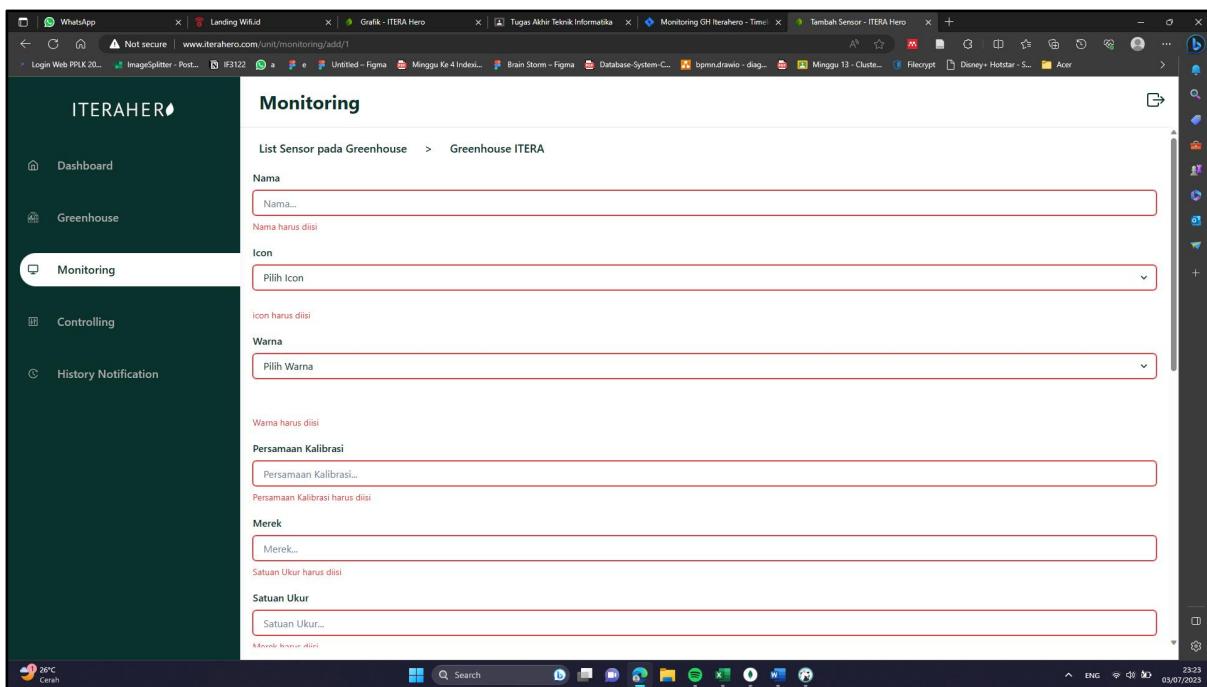
Gambar 4.19 Tampilan Front End halaman monitoring

Pada halaman monitoring jika kita klik tambah pada halaman moitoring akan masuk pada halaman input *sensor* akan masuk ke halaman menambahkan *sensor* berisi mengenai input nama, icon, warna, persamaan kalibrasi, merek, satuan ukur, range max, range min, detail, gambar sensor, dan juga gambar dari posisi sensor pada *greenhouse*. Detail dari halaman penambahan sensor dapat dilihat pada gambar 4.20.



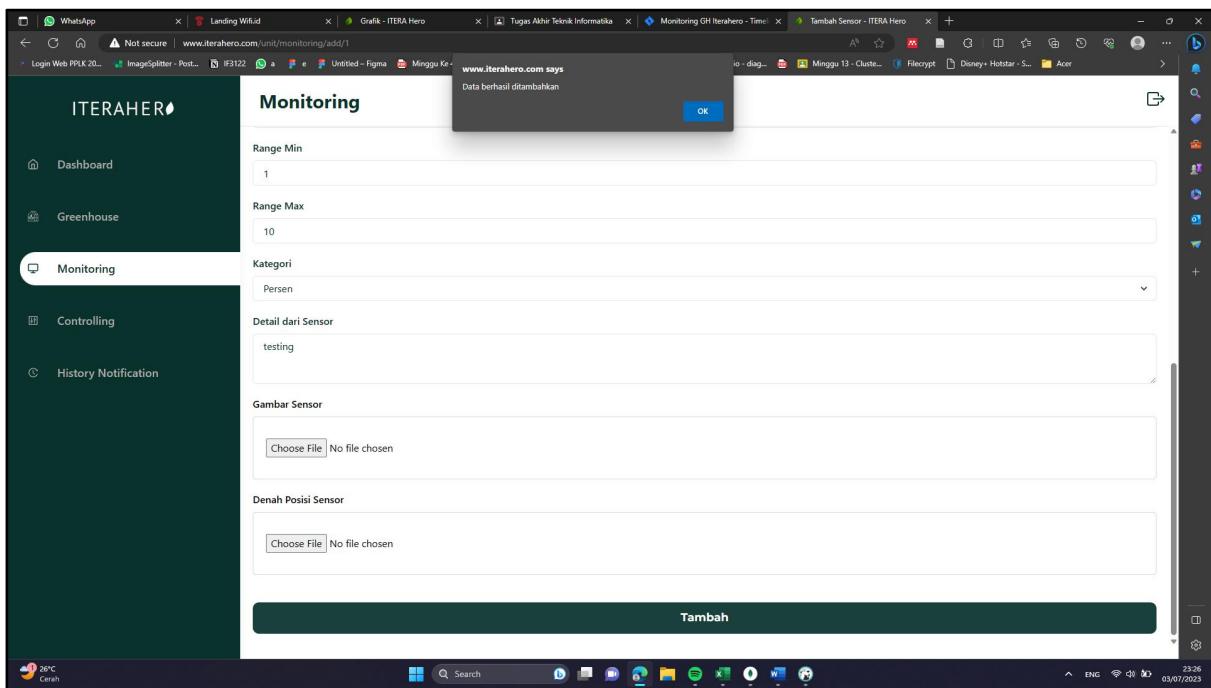
Gambar 4.20 Tampilan menambahkan sensor

Jika kita mengisi list sensor pada *greenhouse* tanpa mengisi data yang ada maka akan *error* dan meminta user untuk tidak mengisi field yang wajib diisi seperti pada gambar 4.21.



Gambar 4.21 Tampilan mensubmit data monitoring tanpa data

Jika semua telah diisi semua maka jika kita klik tambah akan ada pemberitahuan bahwa data yang diisi berhasil seperti pada gambar 4.22.



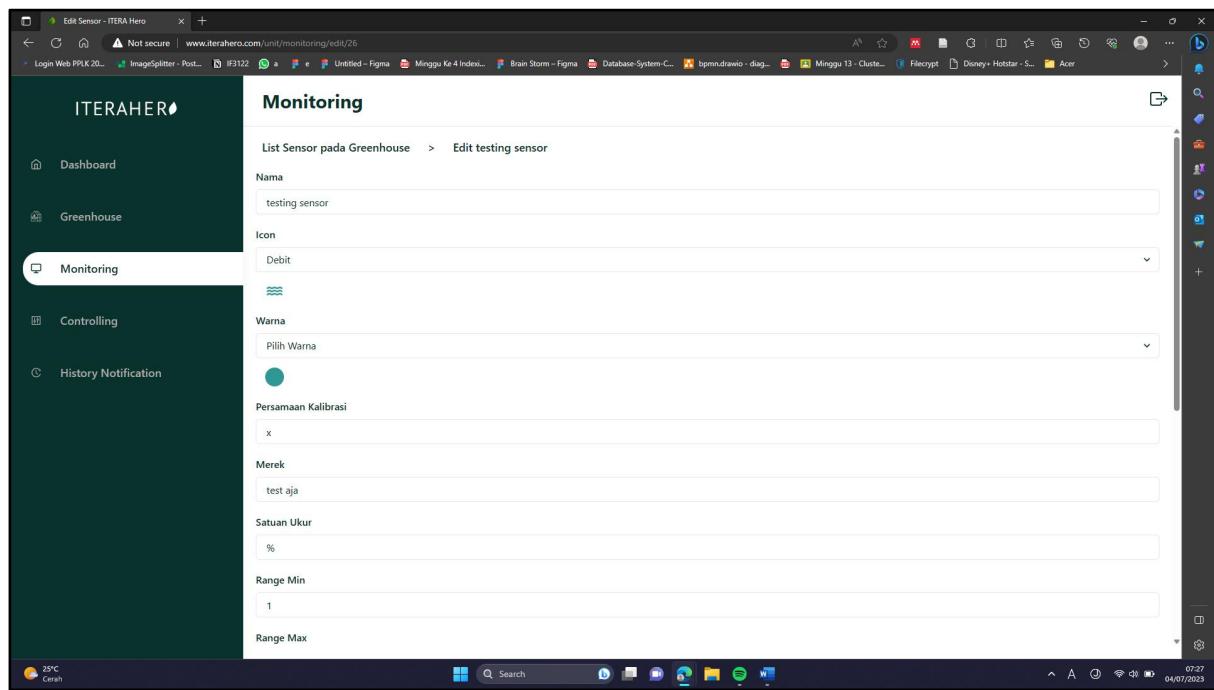
Gambar 4.22 Tampilan jika berhasil menambahkan sensor

Sensor yang telah ditambah jika berhasil akan masuk ke dalam *database* dan menampilkan data yang ada yakni sensor siap seperti pada gambar 4.23.

| NO | NAMA ALAT | ICON | SATUAN UKUR | MEREK | WARNA | PERSAMAAN KALIBRASI | RANGE MIN | RANGE MAX | DETAIL | AKSI |
|----|-----------------------------|------|-------------|----------|-------|---------------------|-----------|-----------|------------------------|---|
| 1 | Suhu Lingkungan Greenhouse | 🌡 | Celcius | DHT11 | ● | x | 27 | 30 | Detail | Edit Delete |
| 2 | Kelembapan Udara Greenhouse | 💧 | % | DHT 11 | ● | x | 70 | 80 | Detail | Edit Delete |
| 3 | Volume Debit Nutrisi | 💧 | ml | YSVS | ● | x*1000-87 | 1 | 1000 | Detail | Edit Delete |
| 4 | testing sensor | 💧 | % | test aja | ● | x | 1 | 10 | Detail | Edit Delete |

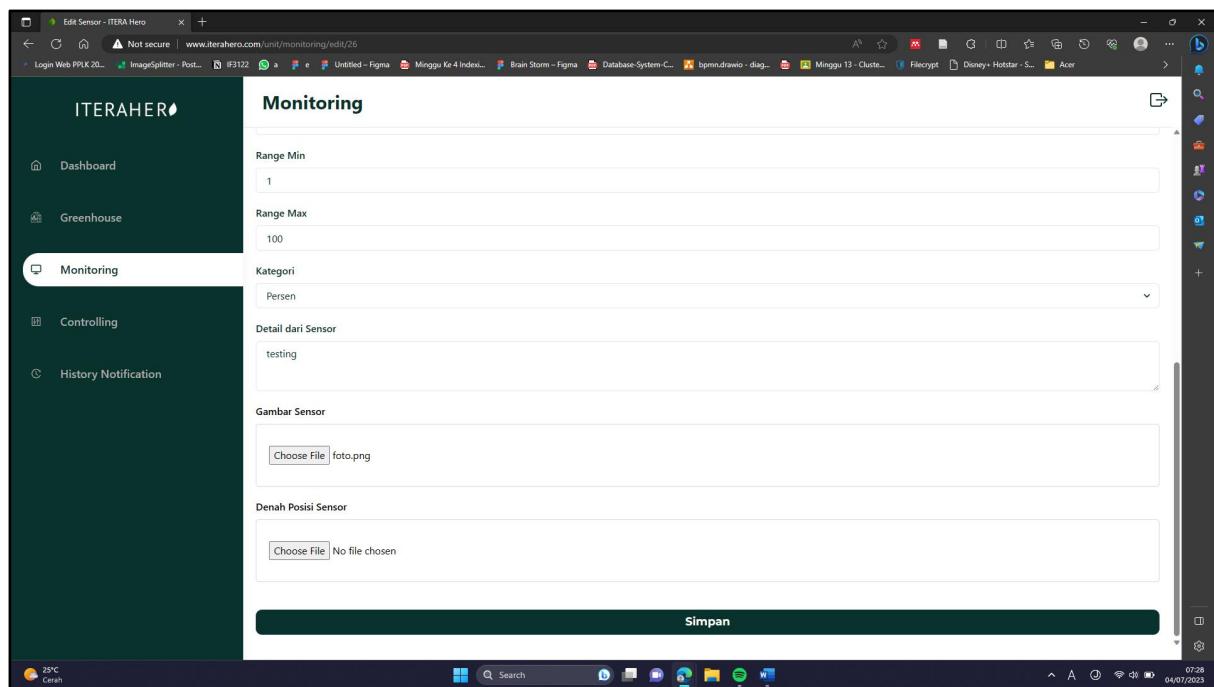
Gambar 4.23 Tampilan setelah menambahkan sensor

Jika dilakukan edit maka akan tampil ke dalam halaman edit, data yang sebelumnya diisi dapat diganti sesuai dengan kebutuhan seperti pada gambar 4.24.



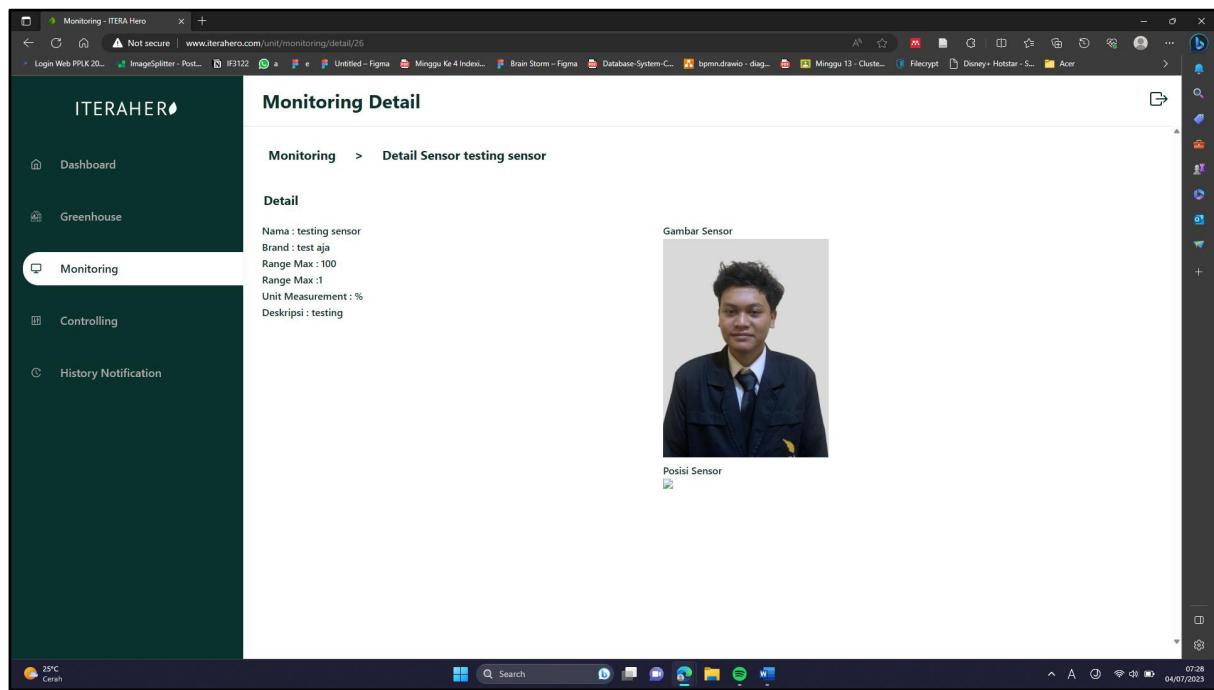
Gambar 4.24 Tampilan front end halaman edit sensor

Jika kita edit data *range max* menjadi 100 dari awalnya 10 dan juga gambar dari sensor seperti pada gambar 4.25.



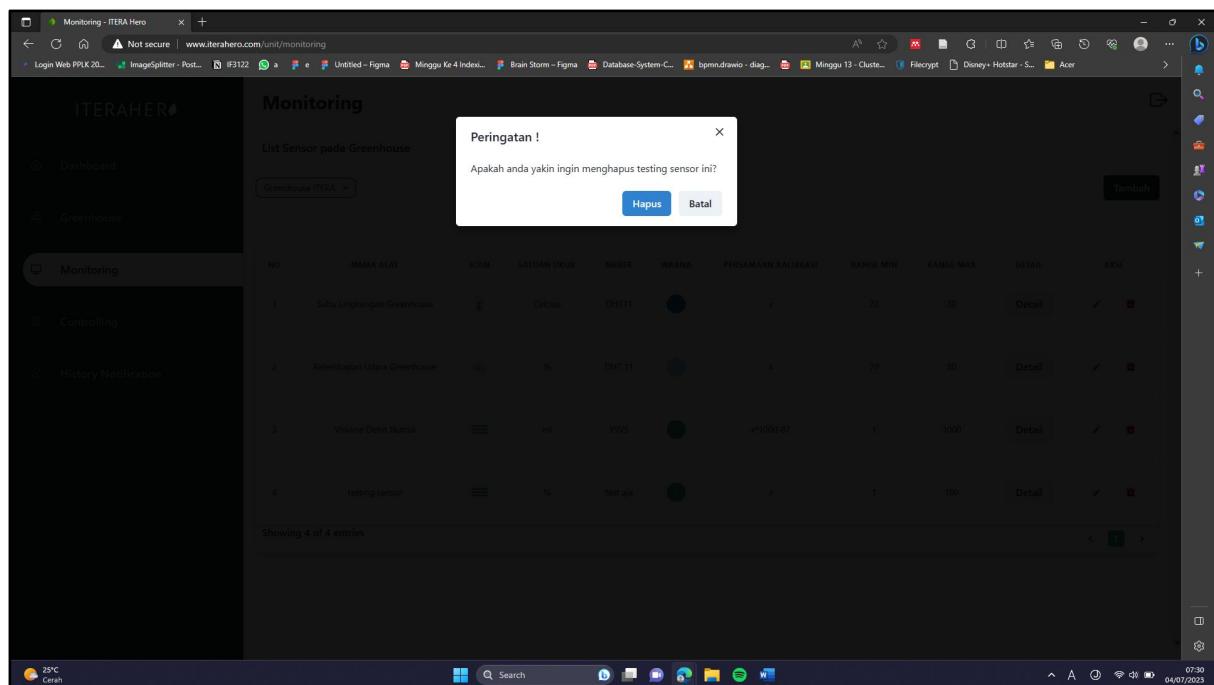
Gambar 4.25 Mengedit range max dan gambar sensor

Setelah dilakukan edit dapat dilihat sensor yang telah diedit maka data yang diedit akan berubah yakni *range max* dan juga gambar dari sensor akan tampil seperti pada gambar 4.26.



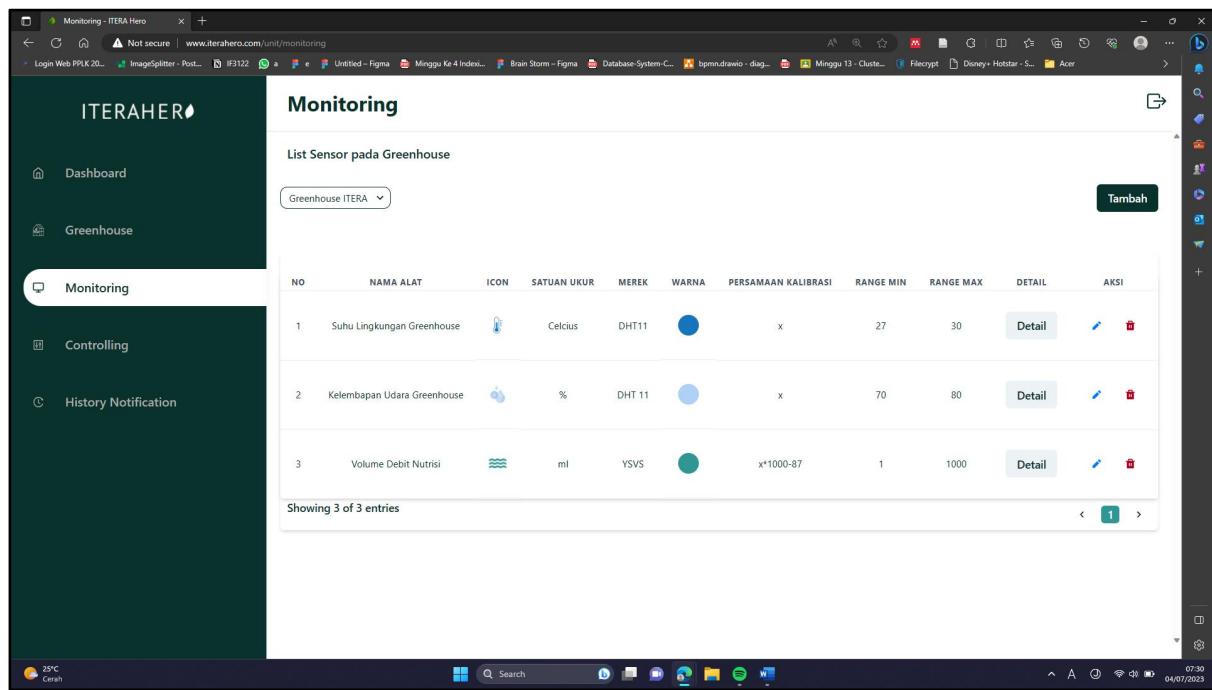
Gambar 4.26 Perubahan setelah sensor dedit

Jika kita menghapus sensor akan ada peringatan untuk melanjutkan menghapus atau batal seperti pada gambar 4.27.



Gambar 4.27 Peringangan menghapus sensor

Sensor yang telah dihapus tidak akan tampil kembali dikarenakan sudah dihapus pada basis data seperti pada gambar 4.28.



Gambar 4.28 Hasil setelah berhasil menghapus sensor

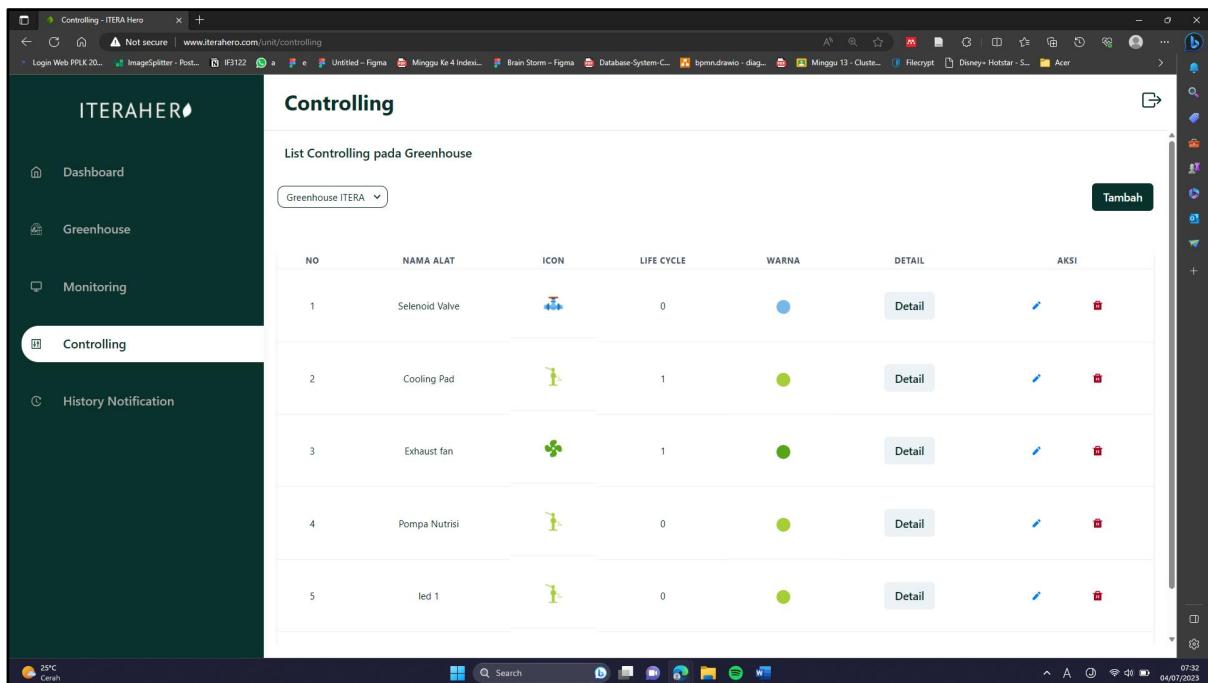
Berdasarkan tampilan sensor yang telah dilakukan maka kelola sistem monitoring berhasil dilakukan mulai dari menampilkan data sesuai *greenhouse*, lalu menambah sensor, mengedit sensor dan menghapus sensor. Rincian dari pengujian yang dilakukan dapat dilihat pada tabel 4.17.

Tabel 4.17 Pengujian black box pada halaman *monitoring*

| Fitur | Case | Indikator Keberhasilan | Hasil |
|--------------------|---|--------------------------|----------|
| Menambah Sensor | Menginput sensor dengan data lengkap | Berhasil menambah sensor | Berhasil |
| | Menginput sensor dengan data yang tidak lengkap | tidak dapat di submit | Berhasil |
| Delete sensor | Menghapus sensor | aktuuator terhapus | Berhasil |
| Edit sensor | Mengedit sensor | aktuuator teredit | Berhasil |
| Menampilkan sensor | Masuk ke halaman sensor dan menampilkan sensor | List sensor tampil | Berhasil |

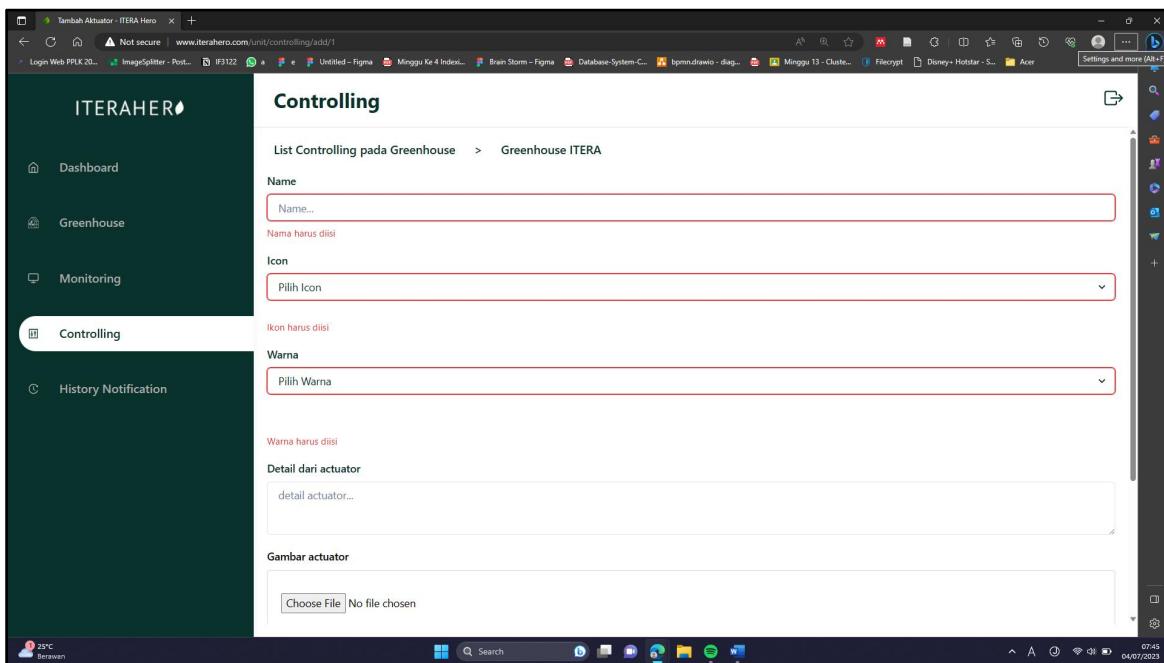
4.1.2.5 Implementasi Halaman Actuator

Pada halaman *controlling* akan menampilkan list dari aktuator yang ada pada *greenhouse* mulai dari nama alat, icon, warna, dan juga detail dari *actuator*, fungsi dari halaman controlling adalah untuk mengelola semua aktuator yang ada pada *greenhouse* sehingga dapat dikendalikan pada halaman *dashboard*, rincian dari pembuatan tampilan *frontend* dapat dilihat hasilnya pada gambar 4.29.



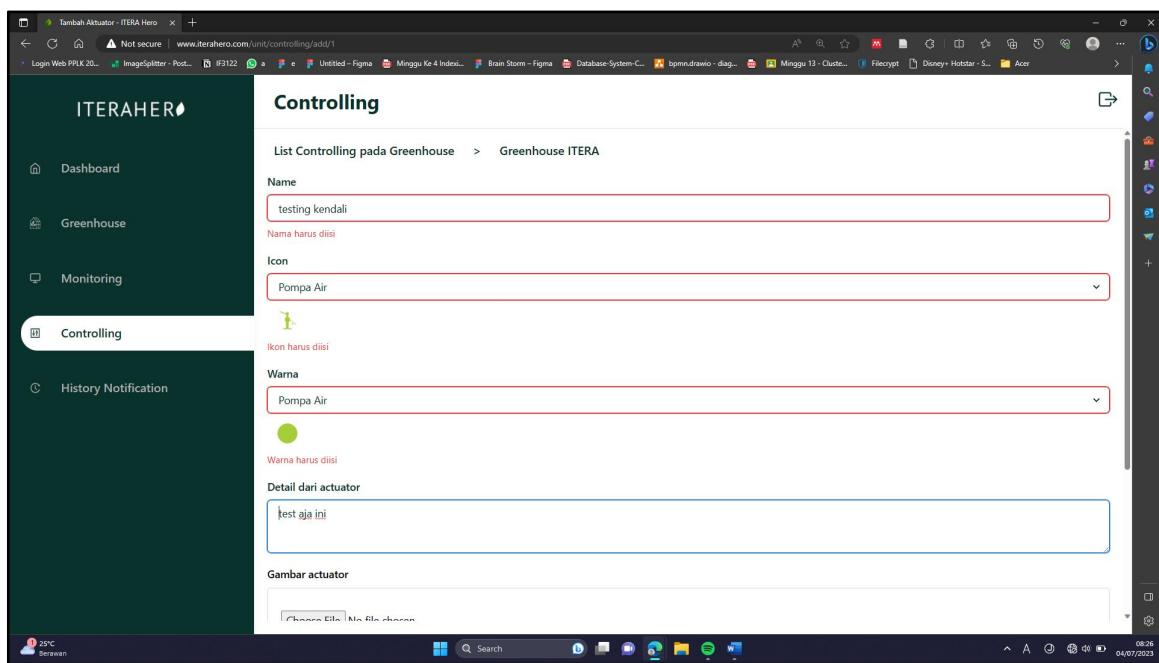
Gambar 4.29 Tampilan front end halaman controlling

Pada halaman tambah maka akan menambah *actuator* mulai dari nama, icon, warna, detail, gambar, denah posisi dari aktuator, dan gambar *aktuator*. Jika tidak mengisi data-data yang perlu diisi maka sistem tidak akan menerima dan ada pemberitahuan untuk mengisi data-data yang dapat dilihat pada gambar 4.30.



Gambar 4.30 Tampilan jika tidak mengisi saat ingin menambah aktuator

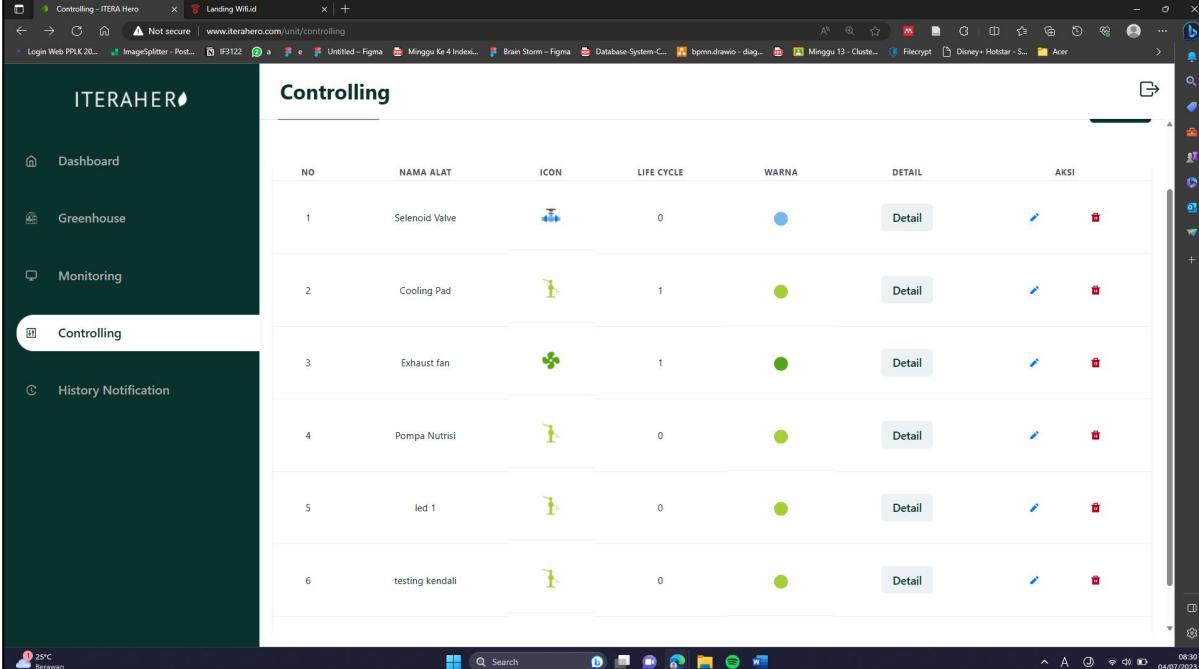
Pada halaman tambah pada *controlling* perlu diisi beberapa data yang perlu yakni seperti nama, icon, warna, dan juga detail dari actuator seperti pada gambar 4.31.



Gambar 4.31 Menginput seluruh data untuk menambah aktuator

Pada halaman tambah pada *controlling* jika klik tambah akan ada pemberitahuan berhasil seperti pada gambar dan kembali. pada halaman halaman *controlling* jika berhasil

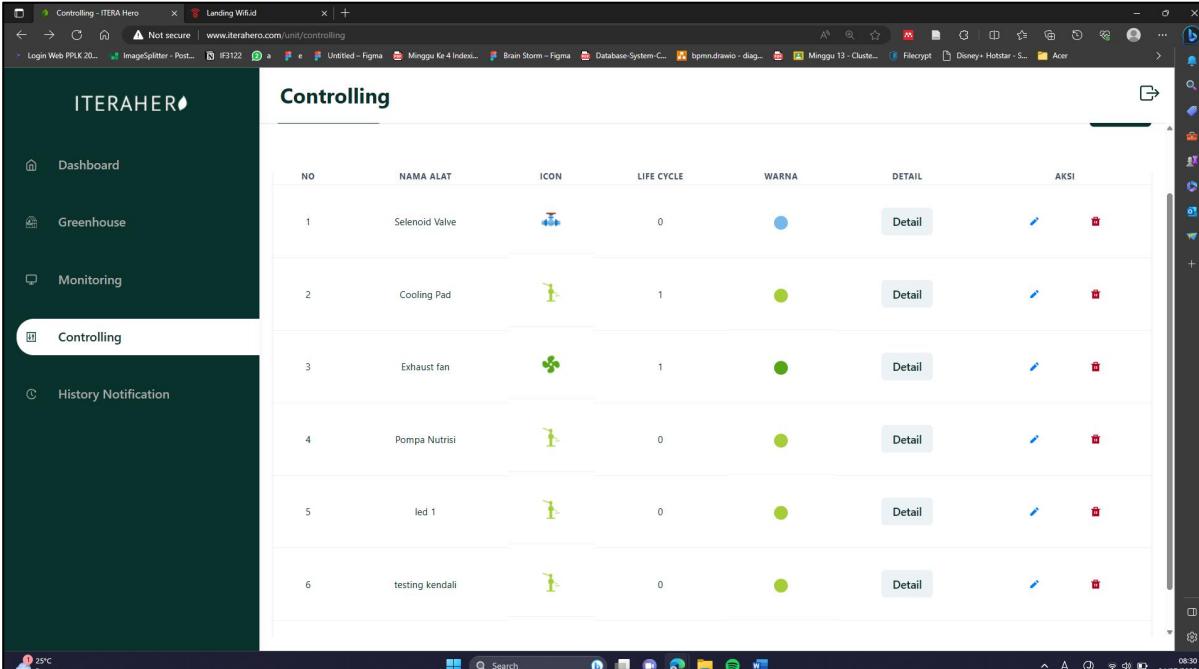
menambahkan data maka akan tampil pada halaman *controlling* data yang sudah ditambah yakni aktuator testing kendali seperti pada gambar 4.32.



| NO | NAMA ALAT | ICON | LIFE CYCLE | WARNA | DETAIL | AKSI |
|----|-----------------|------|------------|-------|------------------------|---|
| 1 | Selenoid Valve | 💡 | 0 | ● | Detail | Edit Delete |
| 2 | Cooling Pad | 💡 | 1 | ● | Detail | Edit Delete |
| 3 | Exhaust fan | 💡 | 1 | ● | Detail | Edit Delete |
| 4 | Pompa Nutrisi | 💡 | 0 | ● | Detail | Edit Delete |
| 5 | led 1 | 💡 | 0 | ● | Detail | Edit Delete |
| 6 | testing kendali | 💡 | 0 | ● | Detail | Edit Delete |

Gambar 4.32 Tampilan setelah menambahkan aktuator

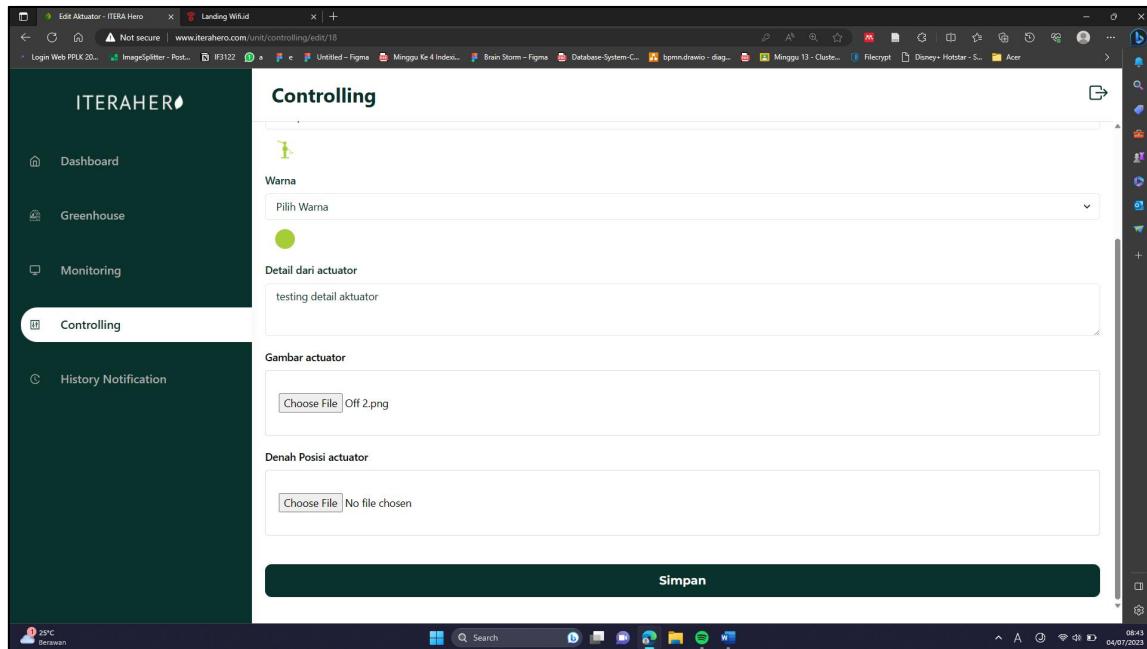
Pada halaman edit berisi mengenai data-data yang telah diisi sebelumnya kita dapat mengganti beberapa data yang telah diisi sebelumnya seperti pada gambar 4.33.



| NO | NAMA ALAT | ICON | LIFE CYCLE | WARNA | DETAIL | AKSI |
|----|-----------------|------|------------|-------|------------------------|---|
| 1 | Selenoid Valve | 💡 | 0 | ● | Detail | Edit Delete |
| 2 | Cooling Pad | 💡 | 1 | ● | Detail | Edit Delete |
| 3 | Exhaust fan | 💡 | 1 | ● | Detail | Edit Delete |
| 4 | Pompa Nutrisi | 💡 | 0 | ● | Detail | Edit Delete |
| 5 | led 1 | 💡 | 0 | ● | Detail | Edit Delete |
| 6 | testing kendali | 💡 | 0 | ● | Detail | Edit Delete |

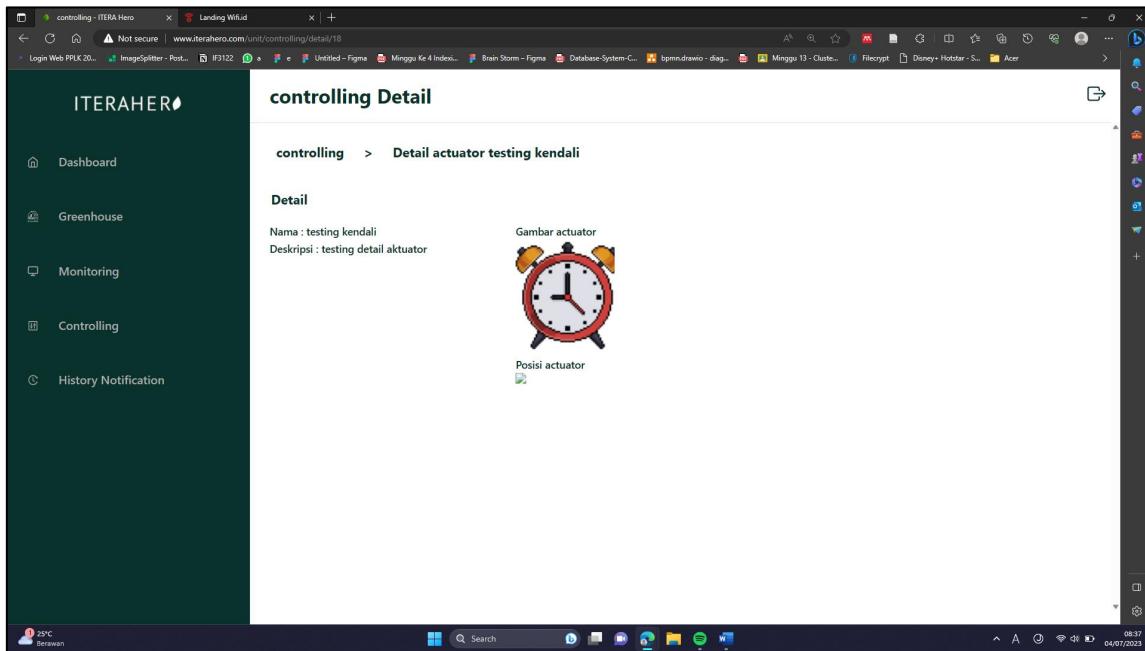
Gambar 4.33 Tampilan front end mengedit aktuator

Jika kita melakukan edit pada actuator testing kendali dengan mengisi detail dan juga gambar dari actuator yang dapat merubah dari detail dan juga gambar, tampilan perubahan dapat dilihat seperti pada gambar 4.34.



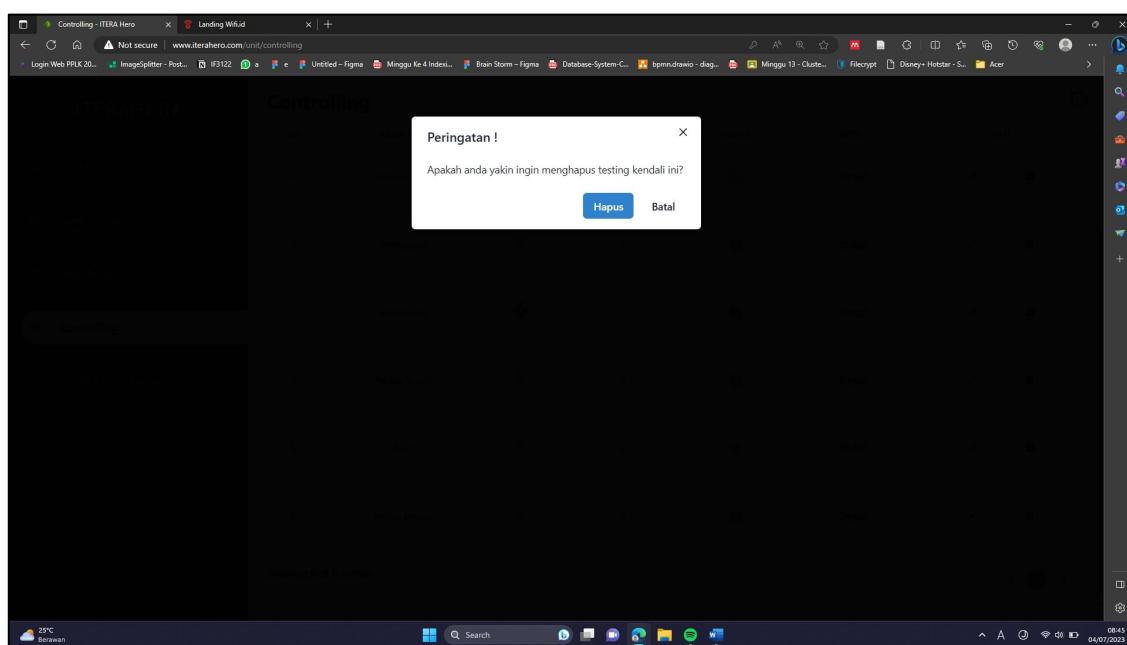
Gambar 4.34 Tampilan front end mengedit aktuator

Data yang telah diedit akan dapat dilihat perubahannya seperti deskripsi menjadi testing detail dan juga gambar dari actuator menjadinya ada *gambarnya* untuk tampilan dapat dilihat pada gambar 4.35.



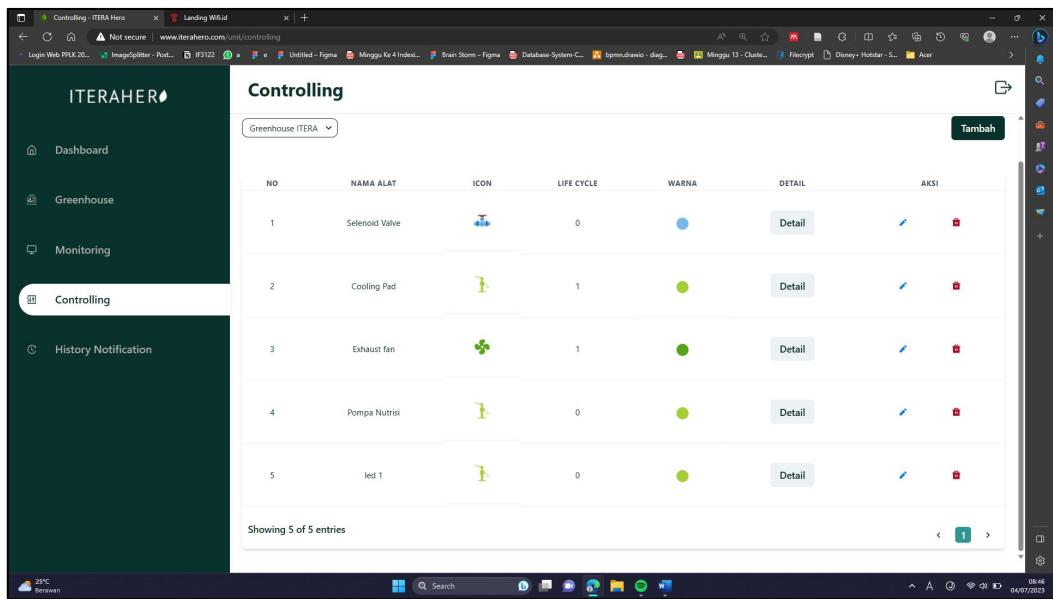
Gambar 4.35 Tampilan hasil mengedit aktuator

Fitur pada *controlling* yakni menghapus aktuator yang terkait jika klik tempat sampah maka akan ada peringatan untuk melanjutkan menghapus atau batal menghapus, jika klik hapus maka akan terhapus dan jika batal akan Kembali ke halaman *controlling*, tampilan dapat dilihat pada gambar 4.34.



Gambar 4.36 Peringatan menghapus aktuator

Hasil dari menghapus maka data aktuator yang dihapus akan hilang dari *website* yakni telah bersail menghapus actuator testing detail sehingga tidak ada lagi tampilan dari hilangnya actuator testing detail dapat dilihat pada gambar 4.37.



Gambar 4.37 Tampiilan setelah menghapus aktuator

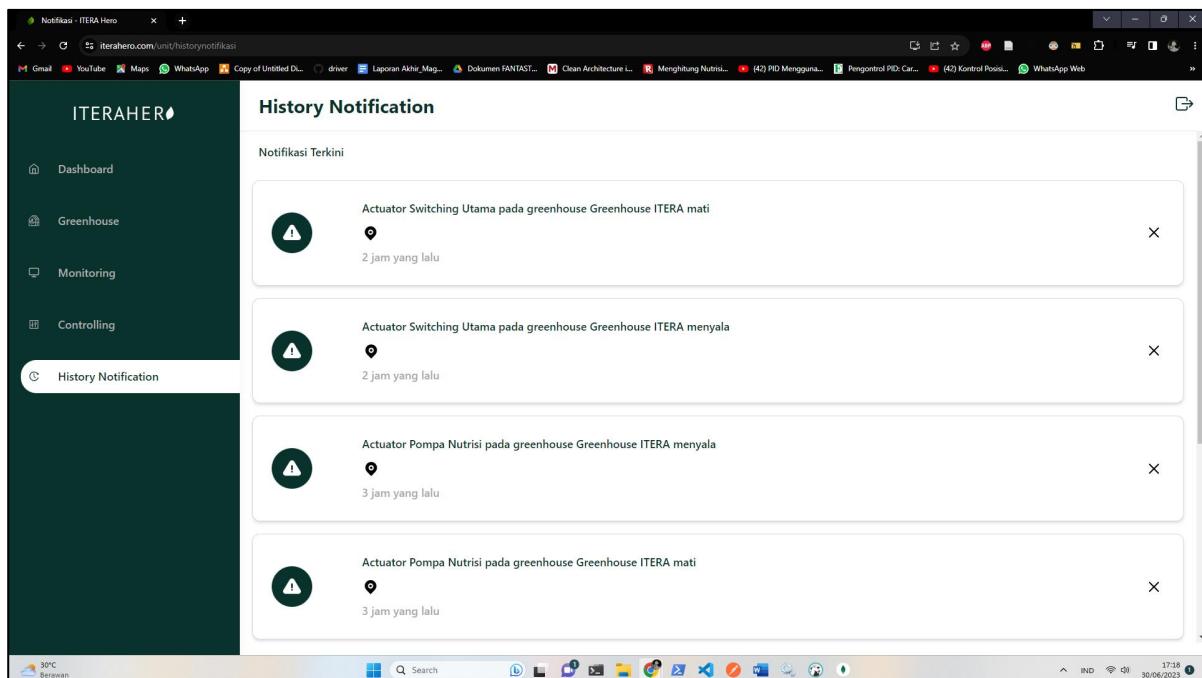
Berdasarkan tampilan *controlling* yang telah dilakukan maka kelola sistem *controlling* berhasil dilakukan mulai dari menampilkan data sesuai *greenhouse*, lalu menambah aktuator, mengedit aktuator dan menghapus aktuator. Rincian dari pengujian yang dilakukan dapat dilihat pada tabel 4.18.

Tabel 4.18 Rincian pengujian black box halaman *controlling*

| Fitur | Case | Indikator Keberhasilan | Hasil |
|----------------------|--|-------------------------------------|----------|
| Menambah Aktuator | Menginput aktuator dengan data lengkap | Berhasil menambah <i>greenhouse</i> | Berhasil |
| | Menginput aktuator dengan data yang tidak lengkap | tidak dapat di submit | Berhasil |
| Delete Aktuator | Menghapus aktuator | aktuator terhapus | Berhasil |
| Edit Aktuator | Mengedit aktuator | aktuator teredit | Berhasil |
| Menampilkan aktuator | Masuk ke halaman aktuator dan menampilkan aktuator | List Aktuator tampil | Berhasil |

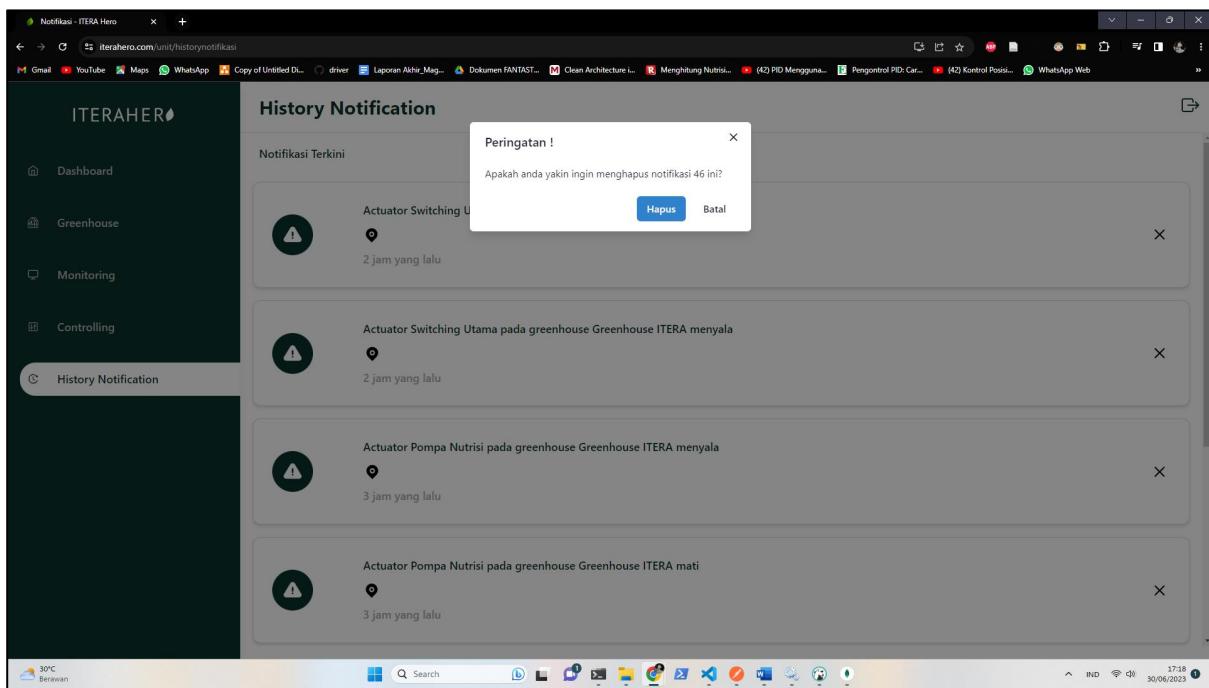
4.1.2.6 Implementasi Halaman Notifikasi

Pada halaman Notifikasi berisi mengenai pemberitahuan yang perlu diberitahukan kepada pengguna dari sistem seperti saat actuator menyala berdasarkan otomatis maka sistem akan menampilkan berhasil menyalakan aktuator dan akan menampilkannya pada halaman notifikasi mulai dari informasi, lokasi, waktu dari notifikasi sehingga pengguna dapat mengetahui keadaaan dari *greenhouse* tampilan dapat dilihat seperti pada gambar 4.38.



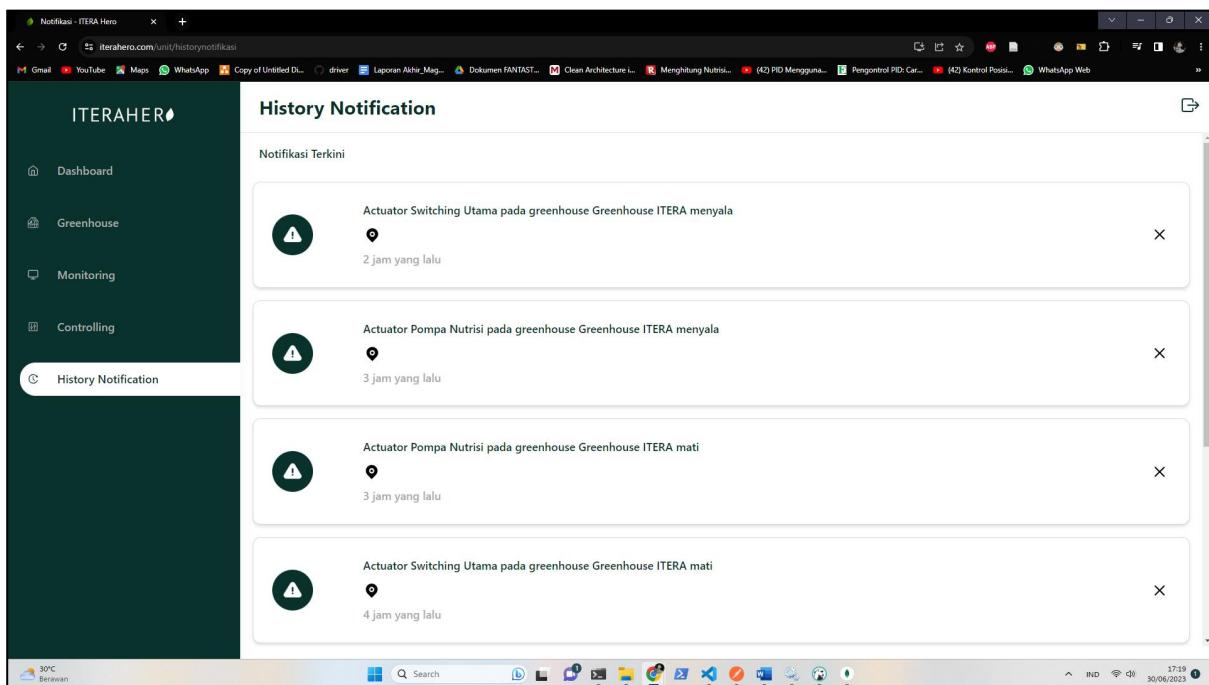
Gambar 4.38 Tampilan front end halaman notifikasi

Pada halaman Notifikasi kita dapat menghapus notifikasi jika notifikasi yang ada terlalu banyak ataupun ada notifikasi yang salah dengan klikhapus akan masuk kedalam *alert* atau pemberitahuan untuk yakin menghapus *notifikasi* yang ada tampilan dapat dilihat pada gambar 4.39.



Gambar 4.39 Peringatan menghapus riwayat notifikasi

Notifikasi akan terhapus jika kita menghapus notifikasi yang ada seperti berhasil menghapus notifikasi nyalanya *switching* utama, tampilan dari hilangnya switing utama setelah pompa nutrisi dapat dilihat pada gambar 4.40.



Gambar 4.40 Tampilan hasil menghapus notifikasi

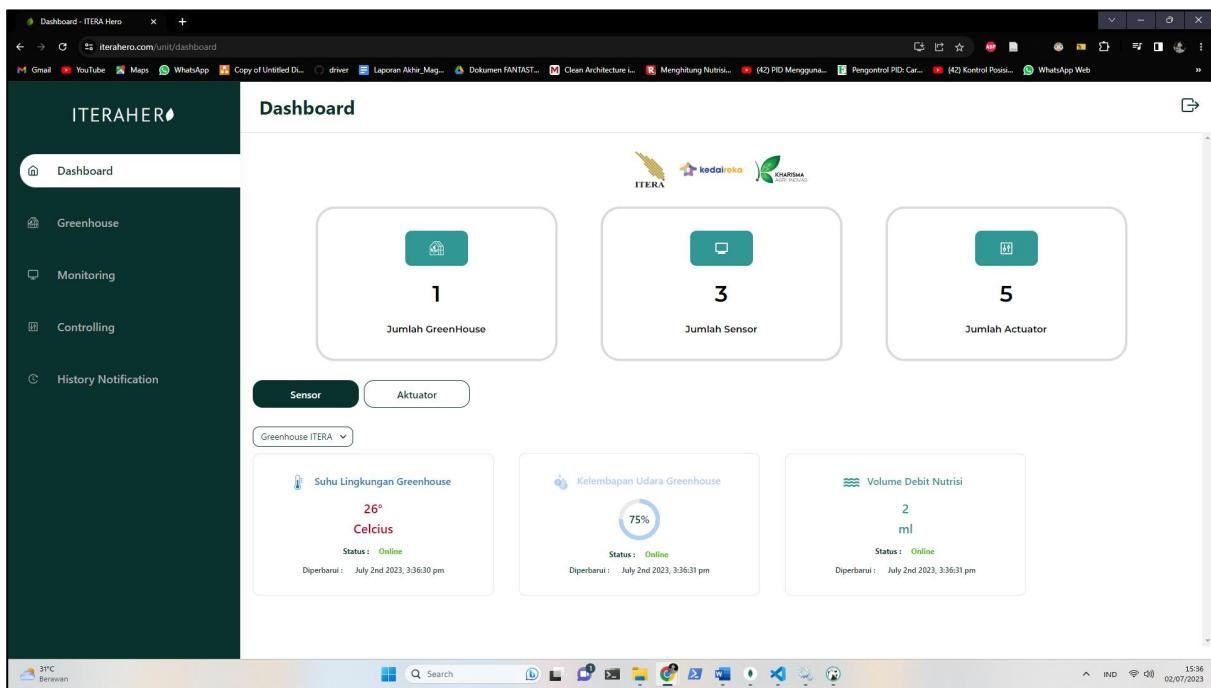
Berdasarkan pengujian yang dilakukan bahwa sistem dapat menampilkan notifikasi dan sistem mampu menghapus notifikasi telah berhasil dilakukan sehingga *blackbox* berhasil apda halaman notifikasi, rincian dari pengujian dapat dilihat pada 4.19.

Tabel 4.19 Rincian pengujian black box halaman notifikasi

| Fitur | Case | Indikator Keberhasilan | Hasil |
|-----------------------------------|--------------------------------|--|----------|
| Sistem memberikan notifikasi | menampilkan halaman notifikasi | Sistem berhasil mendapatkan notifikasi | Berhasil |
| Sistem mampu menghapus notifikasi | menghapus notifikasi | notifikasi terhapus | Berhasil |

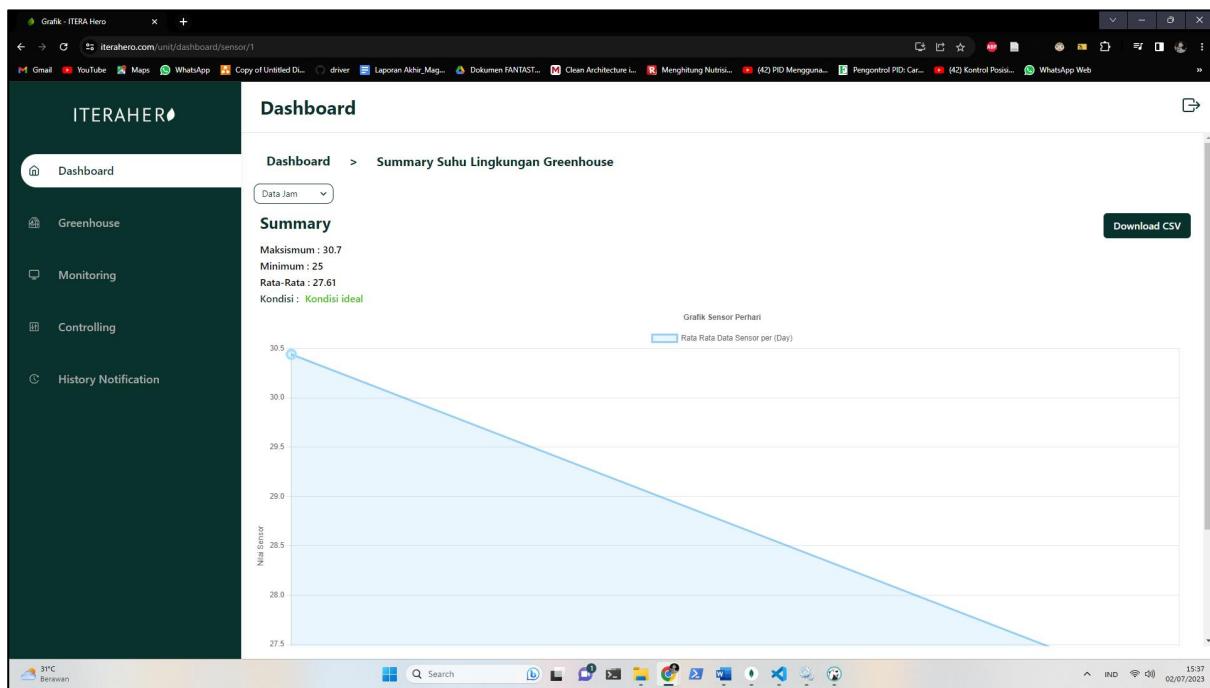
4.1.2.7 Implementasi Halaman Pembacaan Sensor

Halaman pembacaan sensor dapat dilihat pada halaman *dashboard* digunakan untuk menampilkan data-data dari alat ke *website* nilai dari sensor dapat digunakan untuk melakukan *monitoring* dari kondisi yang ada jika warna yang diberikan adalah warna merah mengartikan kondisi yang ada tidak ideal detail dari pembacaan sensor berisi mengenai icon dari sensor, nama sensor, waktu diambilnya data sensor, nilai dari sensor berdasarkan warna yang telah ditentukan, dan juga status dari sensor apakah menyala ataupun mati. tampilan dari *website* untuk menampilkan hasil pembacaan senosr dapat dilihat pada gambar 4.41.



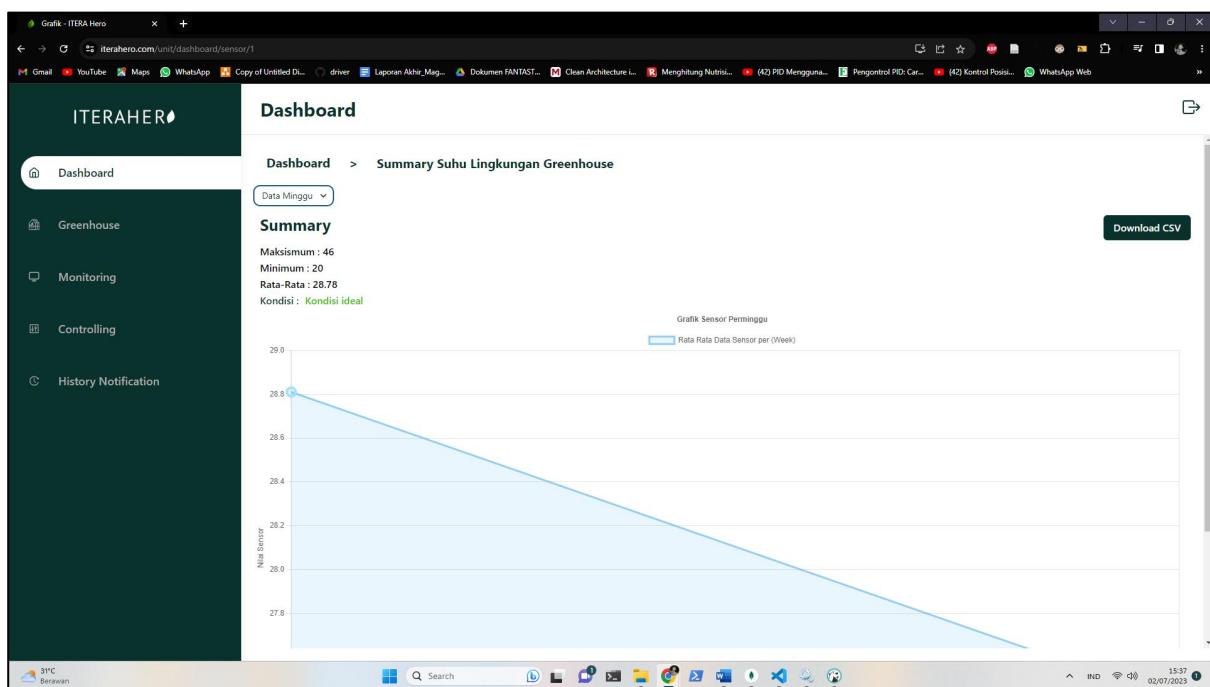
Gambar 4.41 Tampilan front end halaman pembacaan Sensor

Jika kita melakukan klik pada sensor terkait dapat masuk ke dalam detail dari *monitoring* mulai dari nilai terbesar dari sensor, nilai terkecil dari sensor, dan rata-rata pembacaan dari sensor. Serta kondisi berdasarkan pembacaan, detail dari pembacaan dapat dilihat dari grafik dan juga data yang ada dapat di *download* dalam bentuk csv seperti pada gambar 44.2.



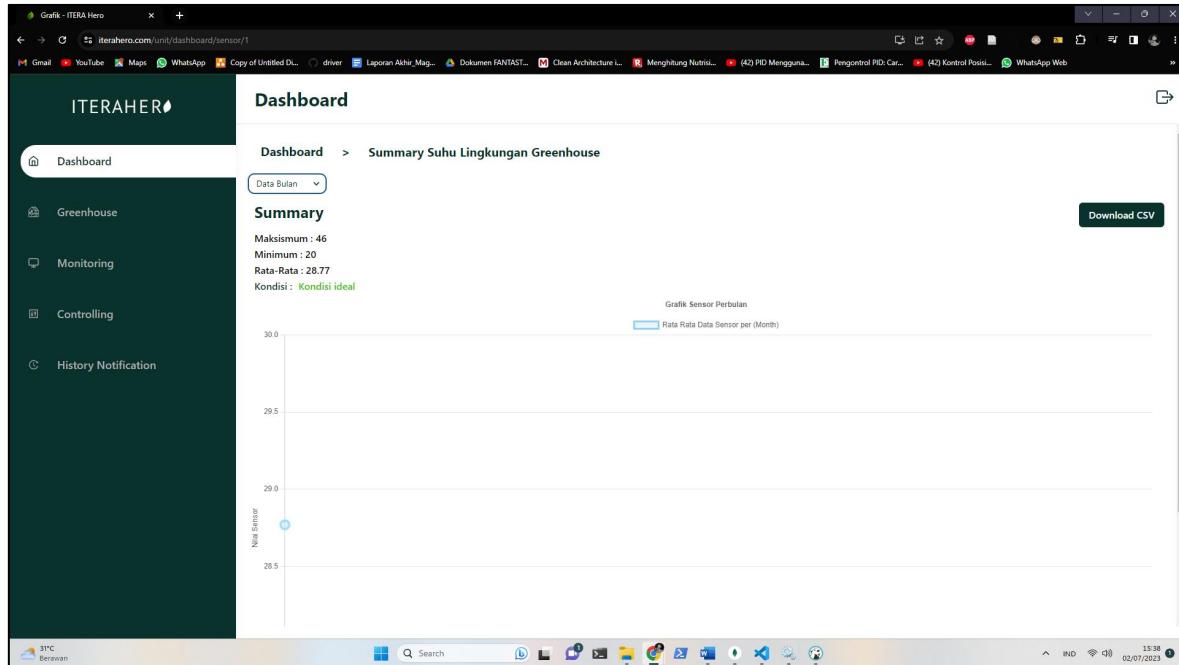
Gambar 4.42 Tampilan front end halaman grafik perjam

Selain data harian adapun data mingguan dari pembacaan sensor yang akan menampilkan data berdasarkan mingguan yakni selama 7 hari akan menampilkan data harian seperti pada gambar 4.43.



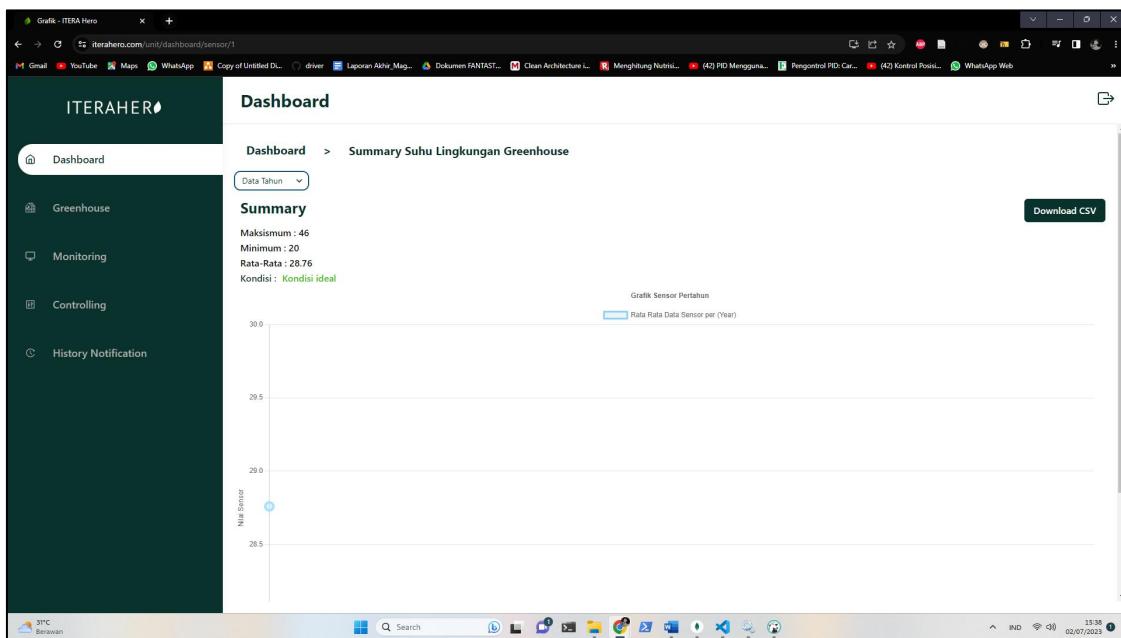
Gambar 4.43 Tampilan front end halaman grafik perminggu

Selain data harian adapun data Bulanan dari pembacaan sensor data ini berisi data mingguan yakni dalam 1 bulan terdapat 4 minggu sehingga minggu 1-4 akan terlihat pada grafik seperti pada gambar 4.44.

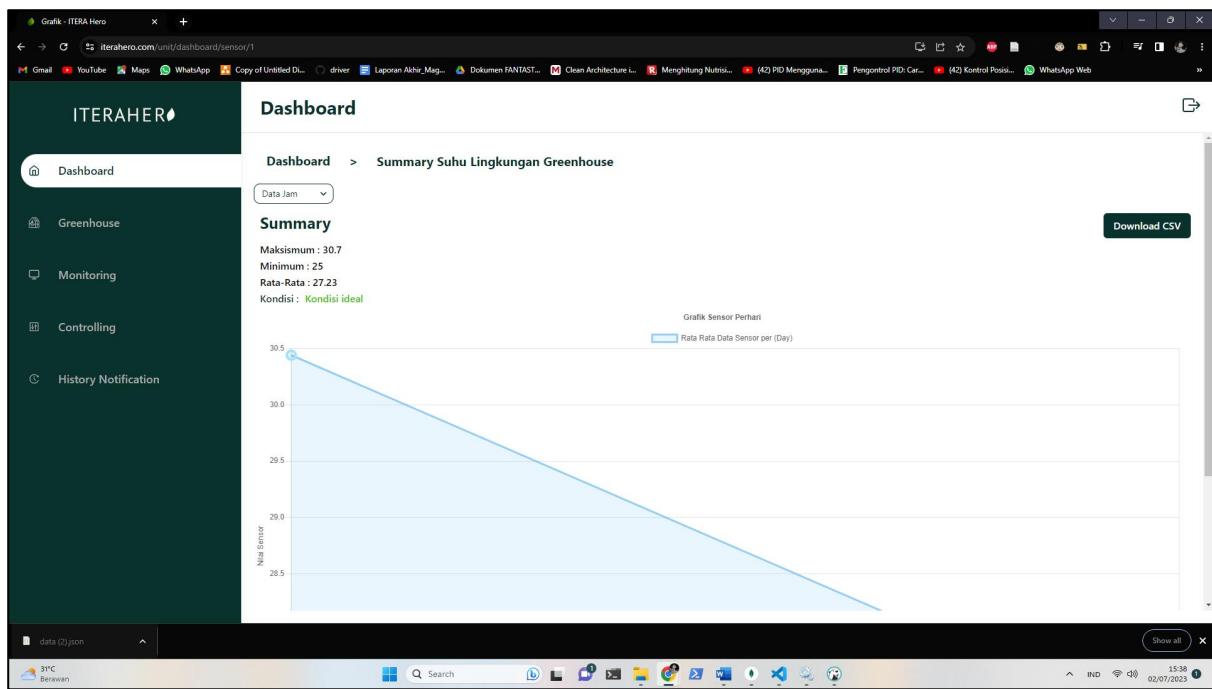


Gambar 4.44 Tampilan front end halaman grafik perbulan

Selain data harian adapun data Tahunan dari pembacaan sensor yang berisi mengenai data berdasarkan bulan 1-12 pada tahun ini yakni pada tahun 2023 Tampilan dapat dilihat dilihat pada gambar 4.45.

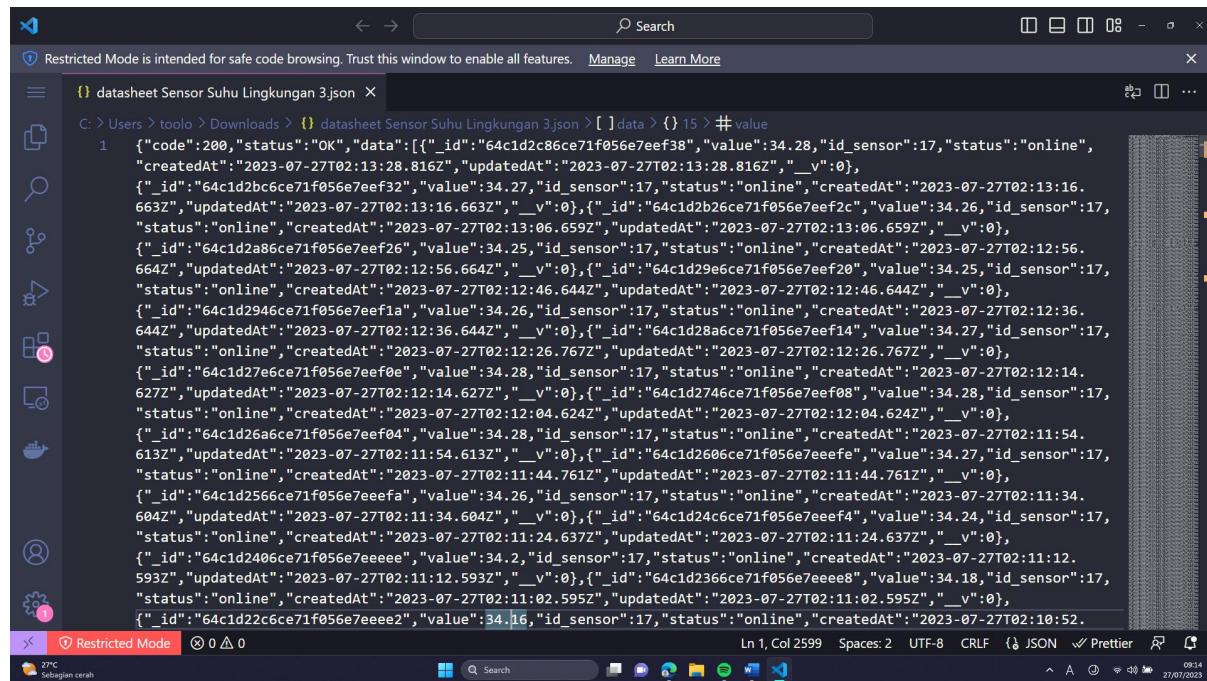


Gambar 4.45 Tampilan front end halaman grafik pertahun
Jika pengguna ingin melihat semua pembacaan data yang ada dapat dengan cara klik *download csv* yan berarti mengunduh seluruh data yang ada dalam bentuk csv, dan sistem akan otomatis mengunduh seperti pada gambar 4.46.



Gambar 4.46 Mengunduh File csv pada data harian

Hasil dari unduhan csv berupa file dalam bentuk csv yang berisi id_sensor, status, dan juga tanggal pembuatan dari data, tampilan dari hasil unduhan csv dapat dilihat pada gambar 4.46.



```

C: > Users > toolo > Downloads > {} datasheet Sensor Suhu Lingkungan 3.json > [ ] data > {} 15 > # value
1   {"code":200,"status":"OK","data":[{"_id":"64c1d2c86ce71f056e7eef38","value":34.28,"id_sensor":17,"status":"online",
  "createdAt":"2023-07-27T02:13:28.816Z","updatedAt":"2023-07-27T02:13:28.816Z","_v":0},
  {"_id":"64c1d2bc6ce71f056e7eef32","value":34.27,"id_sensor":17,"status":"online","createdAt":"2023-07-27T02:13:16.
  663Z","updatedAt":"2023-07-27T02:13:16.663Z","_v":0},{"_id":"64c1d2b26ce71f056e7eef2c","value":34.26,"id_sensor":17,
  "status":"online","createdAt":"2023-07-27T02:13:06.659Z","updatedAt":"2023-07-27T02:13:06.659Z","_v":0},
  {"_id":"64c1d2a86ce71f056e7eef26","value":34.25,"id_sensor":17,"status":"online","createdAt":"2023-07-27T02:12:56.
  664Z","updatedAt":"2023-07-27T02:12:56.664Z","_v":0},{"_id":"64c1d29e6ce71f056e7eef20","value":34.25,"id_sensor":17,
  "status":"online","createdAt":"2023-07-27T02:12:46.644Z","updatedAt":"2023-07-27T02:12:46.644Z","_v":0},
  {"_id":"64c1d2946ce71f056e7eef1a","value":34.26,"id_sensor":17,"status":"online","createdAt":"2023-07-27T02:12:36.
  644Z","updatedAt":"2023-07-27T02:12:36.644Z","_v":0},{"_id":"64c1d28a6ce71f056e7eef14","value":34.27,"id_sensor":17,
  "status":"online","createdAt":"2023-07-27T02:12:26.767Z","updatedAt":"2023-07-27T02:12:26.767Z","_v":0},
  {"_id":"64c1d27e6ce71f056e7eef0e","value":34.28,"id_sensor":17,"status":"online","createdAt":"2023-07-27T02:12:14.
  627Z","updatedAt":"2023-07-27T02:12:14.627Z","_v":0},{"_id":"64c1d2746ce71f056e7eef08","value":34.28,"id_sensor":17,
  "status":"online","createdAt":"2023-07-27T02:12:04.624Z","updatedAt":"2023-07-27T02:12:04.624Z","_v":0},
  {"_id":"64c1d26a6ce71f056e7eef04","value":34.28,"id_sensor":17,"status":"online","createdAt":"2023-07-27T02:11:54.
  613Z","updatedAt":"2023-07-27T02:11:54.613Z","_v":0}, {"_id":"64c1d2606ce71f056e7eef0e","value":34.27,"id_sensor":17,
  "status":"online","createdAt":"2023-07-27T02:11:44.761Z","updatedAt":"2023-07-27T02:11:44.761Z","_v":0},
  {"_id":"64c1d256ce71f056e7eef0a","value":34.26,"id_sensor":17,"status":"online","createdAt":"2023-07-27T02:11:34.
  604Z","updatedAt":"2023-07-27T02:11:34.604Z","_v":0}, {"_id":"64c1d24c6ce71f056e7eef04","value":34.24,"id_sensor":17,
  "status":"online","createdAt":"2023-07-27T02:11:24.637Z","updatedAt":"2023-07-27T02:11:24.637Z","_v":0},
  {"_id":"64c1d2406ce71f056e7eef0e","value":34.22,"id_sensor":17,"status":"online","createdAt":"2023-07-27T02:11:12.
  593Z","updatedAt":"2023-07-27T02:11:12.593Z","_v":0}, {"_id":"64c1d2366ce71f056e7eee08","value":34.18,"id_sensor":17,
  "status":"online","createdAt":"2023-07-27T02:11:02.595Z","updatedAt":"2023-07-27T02:11:02.595Z","_v":0},
  {"_id":"64c1d22c6ce71f056e7eeee2","value":34.16,"id_sensor":17,"status":"online","createdAt":"2023-07-27T02:10:52.
  0934
  27/07/2023
  0 0 0 0
  Restricted Mode
  27C Sebagian cerah
  Ln 1, Col 2599 Spaces: 2 UTF-8 CRLF { JSON Prettier }
```

Gambar 4.47 Hasil Mengunduh file CSV

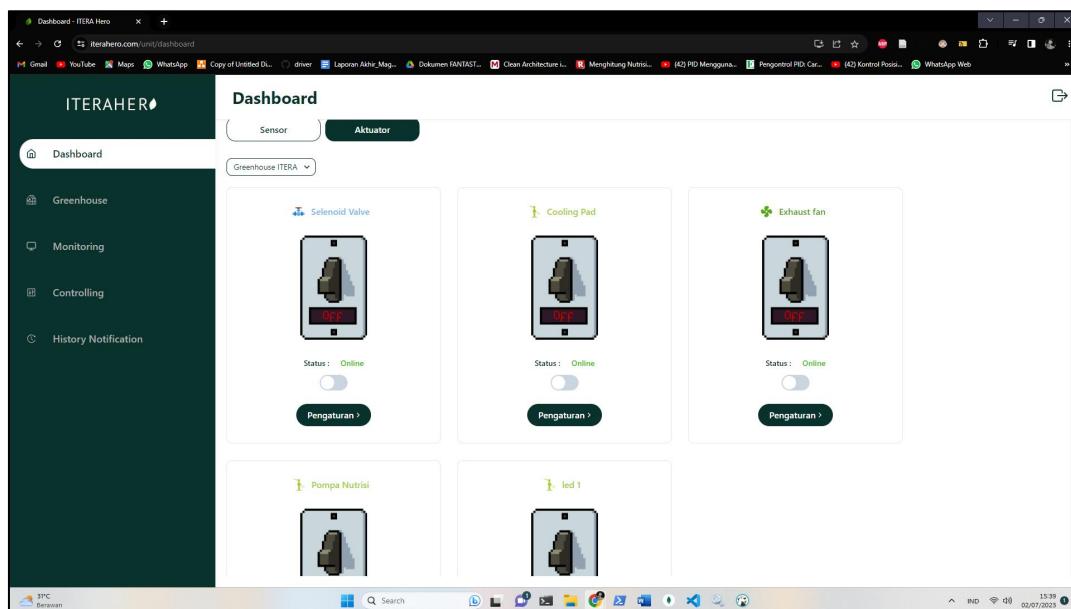
Berdasarkan pengujian yang dilakukan sistem berhasil menampilkan nilai dari sensor mulai dari kondisi sensor dan nilai dari sensor, serta menampilkan grafik, berikut adalah rincian dari pengujian *black box* yang dilakukan 4.20.

Tabel 4.20 Rincian pengujian black box halaman pembacaan sensor

| Fitur | Case | Indikator Keberhasilan | Hasil |
|---------------------------|--|--|----------|
| Menampilkan sensor | Masuk ke halaman sensor dan menampilkan sensor | List sensor tampil | Berhasil |
| Menampilkan Nilai Sensor | Menampilkan sensor pada halaman home | Berhasil menampilkan sensor dan data sensor yang telah dibaca dari perangkat keras | Berhasil |
| Status sensor | Menampilkan status dari sensor | Berhasil menampilkan status sesuai dengan perangkat keras | Berhasil |
| Menampilkan Grafik Sensor | Menampilkan grafik pada detail sensor | Berhasil menampilkan grafik sensor | Berhasil |

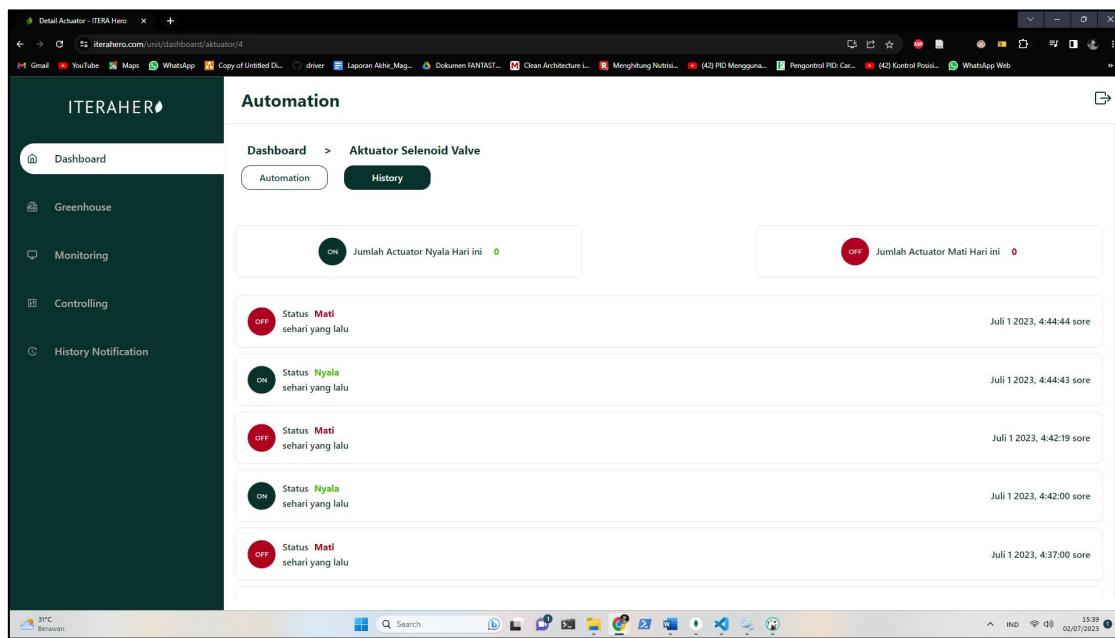
4.1.2.8 Implementasi Halaman Pengendalian Aktuator

Pada halaman *dashboard* terdapat pilihan aktuator dan sensor untuk mengendalikan aktuator yang ada pada *greenhouse* dapat dilakukan melalui *website* terdapat *list* aktuator yang telah dibuat pada halaman *controlling* sehingga kita dapat mengubah status aktuator dari mati menjadi nyala akan tetapi untuk menyalakan dan mematikan memiliki syarat yakni sistem harus *online*. Tampilan dari halaman pengendalian aktuator dapat dilihat pada gambar 4.48.



Gambar 4.48 Tampilan front end halaman pengendalian actuator

Selain kendali aktuator jika kita klik pengaturan akan masuk ke dalam data historis dari aktuator yang menyala dan mati, data yang ada mulai dari hasil perhitungan nyalanya aktuator pada hari ini dan juga mati yang dihitung secara system dan juga untuk data lebih detail adalah terdapat status dari actuator menyala atau mati, lalu waktu dari actuator yang menyala ataupun mati, tampilan dari data historis kendali actuator dapat dilihat pada gambar 4.49.



Gambar 4.49 Tampilan front end data historis status aktuator

Berdasarkan pengujian yang dilakukan sistem berhasil menampilkan status dari aktuator apakah online ataupun offline dan juga riwayat dari nyalanya aktuator, rincian dari pengujian *black box* yang dilakukan dapat dilihat pada tabel 4.21.

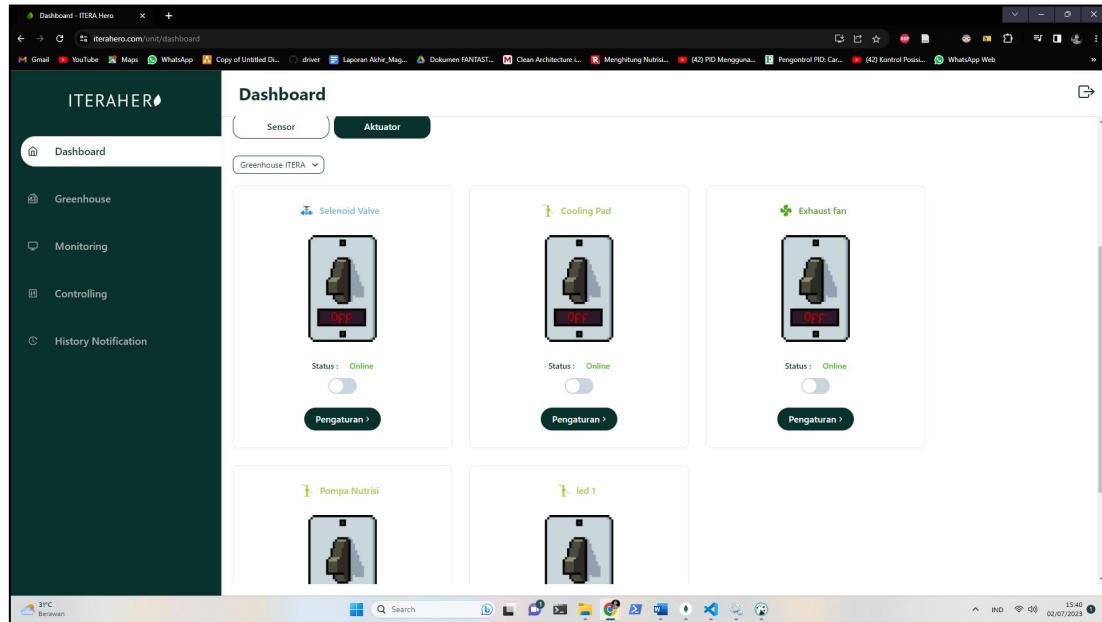
Tabel 4.21 Rincian pengujian black box pada halaman pengendalian aktuator

| Fitur | Case | Indikator Keberhasilan | Hasil |
|------------------|--|--|----------|
| Status aktuator | Sistem mampu menampilkan aktuator offline dan online | Berhasil menampilkan status sesuai dengan perangkat keras | Berhasil |
| riwayat aktuator | menampilkan riwayat aktuator hidup ataupun mati | Sistem mampu menampilkan riwayat aktuator hidup ataupun mati | Berhasil |

4.1.2.9 Implementasi Halaman Pengendalian Otomatis

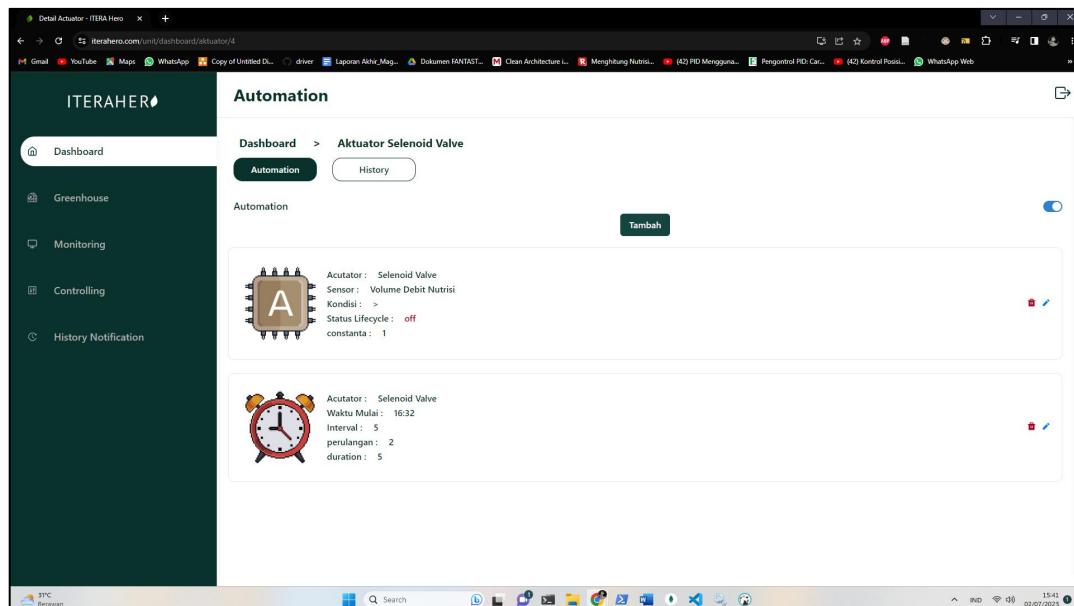
Halaman pengendalian Otomatis digunakan untuk melakukan pengendalian aktuator secara otomatis dapat dilakukan berdasarkan sensor ataupun berdasarkan waktu, untuk

mmengakses otomatis kita perlu masuk ke dalam pengaturan dari aktuator seperti pada gambar 4.50.



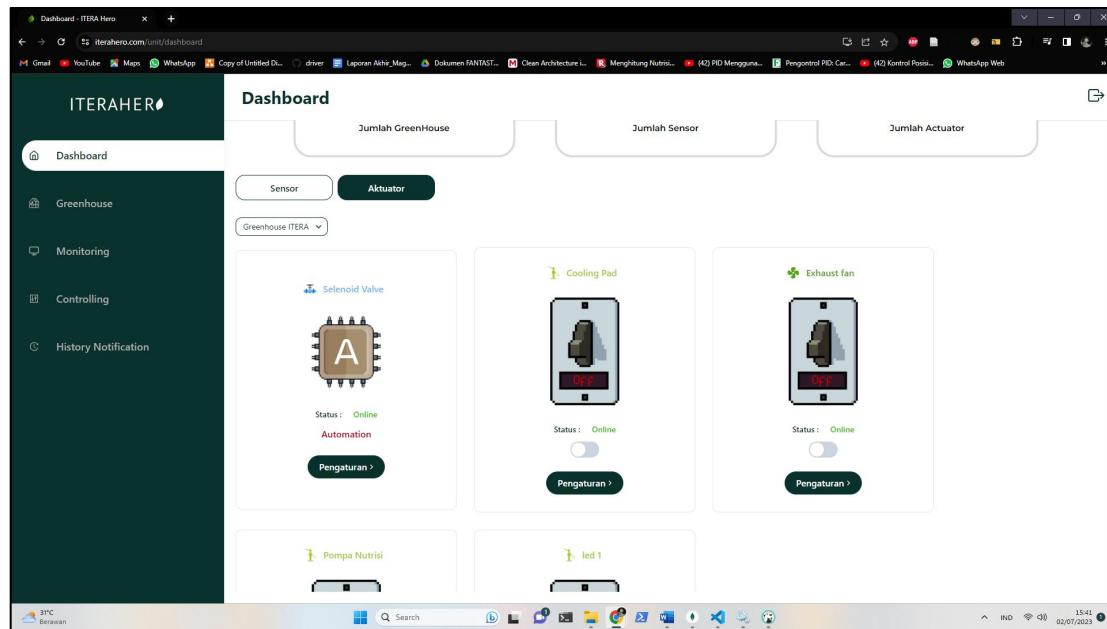
Gambar 4.50 Tampilan front end pengaturan otomatis aktuator

Setelah masuk kedalam halaman pengaturan aktuator kita dapat menambahkan mengedit ataupun menghapus dari otomatisasi sistem yang telah dibuat utuk mengaktifkan otomatis dapat dinyalakan automation dari off ke on seperti pada gambar 4.51.



Gambar 4.51 Tampilan Front end halaman otomatis actuator

Jika kita sudah menghidupkan *automation* maka akan kembali pada aktuator dan icon akan berganti dari manual menjadi otomatis, sehingga pengguna tidak dapat menyalakan ataupun mematikan aktuator, aktuator dikendalikan berdasarkan otomatis yang telah dibuat. Tampilan dari perubahan otomatis dan manual dapat dilihat pada gambar 4.52.



Gambar 4.52 Tampilan Front end jika menyalakan otomatis

Berdasarkan pengujian yang dilakukan sistem berhasil menampilkan aktuator dikontrol berdasarkan sensor, berdasarkan waktu, dan merubah manual menjadi otomatis, berikut adalah rincian dari pengujian *black box* yang dilakukan. Rincian dari pengujian dapat dilihat pada tabel 4.22.

Tabel 4.22 Rincian pengujian black box halaman otomatis

| Fitur | Case | Indikator Keberhasilan | Hasil |
|---------------------|---|---|----------|
| kontrol otomatis | Mengubah kontrol manual jadi otomatis | Sistem berhasil mengubah kontrol manual jadi otomatis | Berhasil |
| kontrol dari sensor | Menambahkan aktuator secara otomatis berdasarkan sensor | Sistem berhasil Menambahkan aktuator secara otomatis berdasarkan sensor | Berhasil |

| Fitur | Case | Indikator Keberhasilan | Hasil |
|--------------------------|--|--|----------|
| kontrol dari penjadwalan | Menambahkan aktuator secara otomatis berdasarkan penjadwalan | Sistem berhasil Menambahkan aktuator secara otomatis | Berhasil |

4.1.3 Impementasi Alat

Setelah website dibangun alat-alat yang digunakan untuk melakukan *monitoring greenhouse* perlu dibuat mulai dari sensor dan juga aktuator. Pada sub bab ini akan menjelaskan hasil dari implementasi alat yang telah dibuat untuk dapat melakukan monitoring dan juga sistem kendali pada *greenhouse*. Hasil dari pembuatan dapat dilihat pada gambar 4.53.



Gambar 4.53 Hasil pembuatan alat pada *Greenhouse* ITERA

4.1.3.1 Sensor DHT (suhu dan Kelembaban)

Sensor dht merupakan sensor yang membaca kelembaban dari *greenhouse* berfungsi untuk monitoring suhu dan juga kelembaban yang ada pada *greenhouse* sensor ini memiliki kasing box berwarna hitam dan terletak di tengah dari *greenhouse* jumlah dari sensor ini adalah 1 buah, untuk tampilan dari sensor DHT 11 dapat dilihat pada gambar 4.54.



Gambar 4.54 Implementasi Sensor DHT 11

Sensor DHT dilakukan pengujian untuk mengukur nilai dari kelembaban dan suhu apakah berhasil menampilkan nilai sensor dari alat ke *website*. Rincian dari pengujian sensor DHT dapat dilihat pada tabel 4.23.

Tabel 4.23 Pengujian nilai sensor DHT11

| No | Jam | Nilai DHT 11 (celcius) | Nilai Alat Ukur (celcius) | Error (persen) |
|----|----------|---------------------------|------------------------------|----------------|
| 1 | 21:43:54 | 32,8 °C | 33 °C | 0,606060606 % |
| 2 | 21:44:04 | 33,1 °C | 33,5 °C | 1,194029851 % |
| 3 | 21:44:14 | 33,5 °C | 34 °C | 1,470588235 % |
| 4 | 21:44:24 | 33,8 °C | 34 °C | 0,588235294 % |
| 5 | 21:44:34 | 34 °C | 34 °C | 0 % |
| 6 | 21:44:44 | 34,4 °C | 34,2 °C | 0,584795322 % |
| 7 | 21:44:54 | 34,5 °C | 34,2 °C | 0,877192982 % |
| 8 | 21:45:04 | 34,8 °C | 34,3 °C | 1,457725948 % |
| 9 | 21:45:14 | 35 °C | 34,5 °C | 1,449275362 % |
| 10 | 21:45:24 | 35,2 °C | 34,5 °C | 2,028985507 % |
| 11 | 21:45:34 | 35,4 °C | 34,8°C | 1,724137931 % |
| 12 | 21:45:44 | 35,5 °C | 35°C | 1,428571429 % |
| 13 | 44:54:00 | 35,6 °C | 35,2 °C | 1,136363636 % |
| 14 | 21:46:04 | 35,8 °C | 35,3 °C | 1,416430595 % |
| 15 | 21:46:14 | 35,9 °C | 35,4 °C | 1,412429379 % |

Berdassarkan pengujian yang dilakukan *error* atau galat dari sensor dht mencapai 1,173955% atau masih mendekati nilai dari alat ukur sehingga, sensor dht jika terjadi *error* yang terlalu besar dilakukan kalibrasi terlebih dahulu akan tetapi *error* yang terjadi tidak

terlalu besar sehingga sensor dapat digunakan untuk melakukan monitoring suhu pada *greenhouse*.

Sensor DHT dilakukan pengujian status dari alat ke *Website* berikut adalah rincian dari pengujian sensor DHT. Hasil pengujian pada tabel 4.24.

Tabel 4.24 Pengujian status kelembaban dan suhu

| No | Status DHT | Status Kelembaban Website | Status Suhu Website | Keterangan |
|----|------------|---------------------------|---------------------|------------|
| 1 | Nyala | Online | Online | berhasil |
| 2 | Mati | Offline | Offline | berhasil |
| 3 | Nyala | Online | Online | berhasil |
| 4 | Mati | Offline | Offline | berhasil |
| 5 | Nyala | Online | Online | berhasil |
| 6 | Mati | Offline | Offline | berhasil |
| 7 | Nyala | Online | Online | berhasil |
| 8 | Mati | Offline | Offline | berhasil |
| 9 | Nyala | Online | Online | berhasil |
| 10 | Mati | Offline | Offline | berhasil |

Berdasarkan pengujian yang dilakukan sistem berhasil menampilkan nilai sensor dari DHT 11 dan status. Rincian pengujian dapat dilihat pada tabel 4.25.

Tabel 4.25 Pengujian blackbox website dan sensor

| Fitur | Case | Indikator Keberhasilan | Hasil |
|--------------------------|--------------------------------------|--|----------|
| Menampilkan Nilai Sensor | Menampilkan sensor pada halaman home | Berhasil menampilkan sensor dan data sensor yang telah dibaca dari perangkat keras | Berhasil |
| Status sensor | Menampilkan status dari sensor | Berhasil menampilkan status sesuai dengan perangkat keras | Berhasil |

4.1.3.2 Sensor Debit

Sensor debit digunakan untuk menghitung debit yang keluar dari pompa nutrisi sehingga pemberian nutrisi dapat diberikan secara tepat tampilan dari sensor ini terletak pada pipa paralon yan berukuran 1 inch dengan sensor yang berbentuk tabung yang berisi dengan

mikrokontroller dan juga komponen untuk mengatur arus. sensor ini berjumlah 1 pada greenhouse. Hasil dari pembuatan sensor Debit dapat dilihat pada gambar 4.55.



Gambar 4.55 Sensor Debit

Pengujian yang dilakukan untuk sensor debit adalah membaca sensor dan hasilnya dapat dilihat dari website. Rincian dari pengujian yang telah dilakukan dapat dilihat pada tabel 4.26.

Tabel 4.26 Pengujian sensor debit

| No | Jam | Sensor Debit (mililiter) | Gelas Ukur (mililiter) | Error (persen) |
|----|-------|--------------------------|------------------------|----------------|
| 1 | 18:28 | 1100 ml | 1113 ml | 1,168014376 % |
| 2 | 18:45 | 1493 ml | 1300 ml | 14,84615385 % |
| 3 | 18:47 | 1463 ml | 1500 ml | 2,466666667 % |
| 4 | 18:57 | 1673 ml | 1600 ml | 4,5625 % |
| 5 | 19:01 | 1643 ml | 1590 ml | 3,333333333 % |
| 6 | 19:02 | 1413 ml | 1400 ml | 0,928571429 % |
| 7 | 19:23 | 1493 ml | 1200 ml | 24,41666667 % |
| 8 | 19:24 | 1923 ml | 1850 ml | 3,945945946 % |
| 9 | 19:27 | 1133 ml | 960 ml | 18,02083333 % |
| 10 | 19:33 | 1583 ml | 1500 ml | 5,533333333 % |
| 11 | 19:35 | 1923 ml | 1850 ml | 3,945945946 % |
| 12 | 19:37 | 1663 ml | 1550 ml | 7,290322581 % |
| 13 | 19:39 | 3153 ml | 2950 ml | 6,881355932 % |
| 14 | 19:40 | 3153 ml | 2990 ml | 5,451505017 % |
| 15 | 19:44 | 1543 ml | 1450 ml | 6,413793103 % |

Hasil pengujian didapatkan *error* atau galat yang terjadi adalah 7,28% hal ini disebabkan berasal dari beberapa faktor jika sensor debit melebihi ukuran dari gelas ukur dikarenakan nutrisi pada tandon tidak banyak, sedangkan pada sensor debit yang lebih banyak dikarenakan volume pada tandon penuh, dan pembacaan debit masih terjadi walaupun *selenoid valve* masih menyala sehingga posisi dari debit dan *selenoid* dapat mempengaruhi pembacaan sensor dan hasil dari gelas ukur. Selain itu terjadi permasalahan *error* jika debit yang dikeluarkan sedikit sehingga dapat disimpulkan semakin banyak debit yang mengalir dapat mempengaruhi galat dari sensor debit.

Pengujian yang dilakukan untuk sensor debit lainnya adalah kondisi dari alat apakah kondisi alat sesuai dari kondisi aktual dan status dari *website*. Rincian dari pengujian yang telah dilakukan dapat dilihat pada tabel 4.27.

Tabel 4.27 Pengujian status alat dan website sensor debit

| No | Status Sensor Debit | Status Website | Hasil |
|----|---------------------|----------------|----------|
| 1 | Nyala | Online | Berhasil |
| 2 | Mati | Offline | Berhasil |
| 3 | Nyala | Online | Berhasil |
| 4 | Mati | Offline | Berhasil |
| 5 | Nyala | Online | Berhasil |
| 6 | Mati | Offline | Berhasil |
| 7 | Nyala | Online | Berhasil |
| 8 | Mati | Offline | Berhasil |
| 9 | Nyala | Online | Berhasil |
| 10 | Mati | Offline | Berhasil |

Berdasarkan pengujian yang dilakukan dapat disimpulkan sistem sensor dan *website* berhasil sesuai dengan respon yang diinginkan.

Pengujian yang dilakukan untuk sensor debit adalah membaca sensor dan hasilnya dapat dilihat dari *website* dan status offline online. Sehingga berdasarkan hasil tersebut pengujian berhasil. Rincian dari pengujian yang telah dilakukan dapat dilihat pada tabel 4.28.

Tabel 4.28 Rincian pengujian black box website dan sensor debit

| Fitur | Case | Indikator Keberhasilan | Hasil |
|--------------------------|--------------------------------------|--|----------|
| Menampilkan Nilai Sensor | Menampilkan sensor pada halaman home | Berhasil menampilkan sensor dan data sensor yang telah dibaca dari perangkat keras | Berhasil |

| | | | |
|---------------|--------------------------------|---|----------|
| Status sensor | Menampilkan status dari sensor | Berhasil menampilkan status sesuai dengan perangkat keras | Berhasil |
|---------------|--------------------------------|---|----------|

4.1.3.3 Aktuator Selenoid Valve

Aktuator dari selenoid valve adalah mengontrol jumlah dari nutrisi yang diinginkan fungsinya seperti keran sehingga jumlah nutrisi dapat dilakukan secara otomatis dari sistem, bentuk dari *selenoid valve* adalah konektor pipa dengan ukuran 1 inch dan untuk tabung yang ada pada *solenoid valve* berisi *mikrokontroller* sehingga *solenoid* dapat diatur dari internet dan juga dapat menyala. Untuk solenoid valve berjumlah 1 pada *greenhouse* dengan letak pada pipa paralon berukuran 1 inch pada pipa sebelum nutrisi sampai pada tanaman melon, fungsi dari *selenoid valve* adalah melakukan pemberian nutrisi secara otomatis sehingga mengurangi pengguna yang lupa untuk memberikan nutrisi pada tanaman. Tampilan dari aktuator *selenoid valve* dapat dilihat pada gambar 4.56.



Gambar 4.56. Hasil Pembuatan *Selenoid Valve*

Pengujian pertama yang dilakukan adalah menguji *selenoid valve* apakah input dari website sesuai dengan kondisi dari *selenoid valve*. Rincian dari pengujian yang dilakukan pada *selenoid valve* dapat dilihat pada tabel 4.28.

Tabel 4.29 Pengujian pengendalian *solenoid valve* melalui *website*

| No | Input pada Website | Kondisi Selenoid Valve | Keterangan |
|----|--------------------|------------------------|------------|
| 1 | On | Nyala | Berhasil |
| 2 | Off | Mati | Berhasil |
| 3 | On | Nyala | Berhasil |
| 4 | Off | Mati | Berhasil |
| 5 | On | Nyala | Berhasil |
| 6 | Off | Mati | Berhasil |
| 7 | On | Nyala | Berhasil |
| 8 | Off | Mati | Berhasil |
| 9 | On | Nyala | Berhasil |
| 10 | Off | Mati | Berhasil |
| 11 | On | Nyala | Berhasil |
| 12 | Off | Mati | Berhasil |
| 13 | On | Nyala | Berhasil |
| 14 | Off | Mati | Berhasil |
| 15 | On | Nyala | Berhasil |
| 16 | Off | Mati | Berhasil |

Berdasarkan pengujian terdapat input dan juga hasil dari kondisi aktual *solenoid valve* telah sesuai sehingga respon dari *website* pada alat telah sesuai.

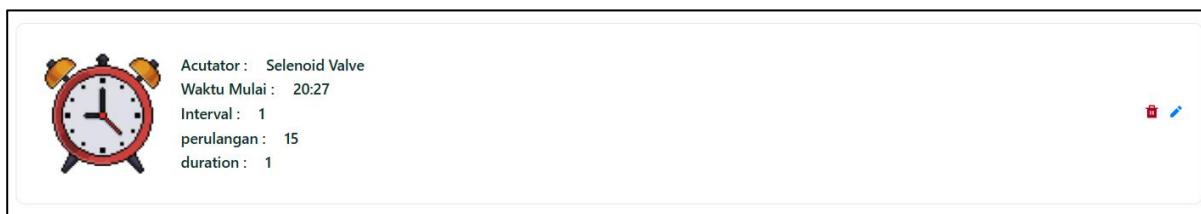
Pengujian kedua yang dilakukan adalah menguji *solenoid valve* apakah kondisi *offline* dan *online* sesuai dari *website*. Rincian dari pengujian yang dilakukan pada *solenoid valve* dapat dilihat pada tabel 4.30.

Tabel 4.30 Pengujian kondisi *solenoid valve* aktual dan *website*

| No | Status Alat Selenoid Valve | Status Website | Keterangan |
|----|----------------------------|----------------|------------|
| 1 | Nyala | Online | Berhasil |
| 2 | Mati | Offline | Berhasil |
| 3 | Nyala | Online | Berhasil |
| 4 | Mati | Offline | Berhasil |
| 5 | Nyala | Online | Berhasil |
| 6 | Mati | Offline | Berhasil |
| 7 | Nyala | Online | Berhasil |
| 8 | Mati | Offline | Berhasil |
| 9 | Nyala | Online | Berhasil |
| 10 | Mati | Offline | Berhasil |

Berdasarkan pengujian terdapat kondisi aktual *solenoid valve* dan kondisi menyala alat pada *website* telah sesuai sehingga kondisi *offline* dan *online* dari *website* dan *solenoid valve* telah sesuai.

Pengujian terakhir adalah menguji otomatis yang dilakukan adalah menguji *solenoid valve* dengan menginputkan otomatis dengan waktu mulai 16:32 dengan durasi menyala 5 menit dan perulangan dilakukan sebanyak 2 kali dimana interval antar alat menyala 5 menit. Tampilan input otomatis dapat dilihat pada gambar 4.57.



Gambar 4.57 Rincian otomatis solenoid valve berdasarkan waktu

Selain berdasarkan penjadwalan adapun otomatis berdasarkan sensor terdapat rincian otomatis *solenoid valve* berdasarkan sensor debit jika melebihi dari *range max* pada sensor debit diset pada *range max* 1000 ml sehingga sistem akan meminta *solenoid valve* untuk mati jika sensor debit membaca lebih dari 1000 ml. Rincian dari otomatis dapat dilihat 4.58.



Gambar 4.58 Rincian otomatis solenoid valve berdasarkan sensor debit

Berdasarkan pengujian yang telah dilakukan dapat diketahui bahwa sistem dapat mematikan dan menyalakan aktuator, lalu dapat menampilkan status dari aktuator telah berhasil. Rincian dari pengujian yang dilakukan dapat dilihat pada tabel 4.31.

Tabel 4.31 Pengujian otomatis *solenoid valve*

| No | Waktu Nyala | Waktu Mati | Jumlah seharusnya (mililiter) | Jumlah sensor debit (mililiter) |
|----|-------------|------------|-------------------------------|---------------------------------|
| 1 | 19:50:02 | 19:50:11 | 1000 ml | 1473 ml |
| 1 | 19:52:04 | 19:52:14 | 1000 ml | 1443 ml |
| 2 | 19:54:01 | 19:54:10 | 1000 ml | 1543 ml |
| 3 | 19:56:02 | 19:56:11 | 1000 ml | 1653 ml |
| 4 | 19:58:02 | 19:58:11 | 1000 ml | 1753 ml |
| 5 | 20:00:01 | 20:00:10 | 1000 ml | 1543 ml |
| 6 | 20:03:02 | 20:03:11 | 1000 ml | 1563 ml |
| 7 | 20:06:01 | 20:06:11 | 1000 ml | 1893 ml |
| 8 | 20:08:02 | 20:08:12 | 1000 ml | 1593 ml |
| 9 | 20:10:01 | 20:10:12 | 1000 ml | 1763 ml |
| 10 | 20:12:02 | 20:12:10 | 1000 ml | 1433 ml |
| 11 | 20:14:01 | 20:14:11 | 1000 ml | 1593 ml |
| 12 | 20:16:02 | 20:16:11 | 1000 ml | 1453 ml |
| 13 | 20:18:01 | 20:18:10 | 1000 ml | 1533 ml |
| 14 | 20:20:01 | 20:20:10 | 1000 ml | 1573 ml |

Terjadi perbedaan yang cukup besar pada otomatis jumlah seharusnya dengan jumlah yang dikeluarkan oleh pompa nutrisi hal ini disebabkan pembacaan delay yang terlalu besar dari sensor debit dan juga terdapat delay saat masuk ke server dan terakhir perintah dari otomatis berdasarkan sensor terdapat delay. Akan tetapi pada penjadwalan waktu delay dan penjadwalan di rentan 1 sampai 2 detik akan tetapi terdapat masalah jika setelah pompa dan selenoid valve mati tetapi air tetap mengalir.

Berdasarkan pengujian yang telah dilakukan dapat disimpulkan bahwa pengujian *solenoid valve* berhasil dilaksanakan rincian dari pengujian blackbox dapat dilihat pada tabel 4.32.

Tabel 4.32 Rincian pengujian aktuator *solenoid valve* dan website

| Fitur | Case | Indikator Keberhasilan | Hasil |
|--|---|--|----------|
| Sistem dapat mematikan dan menyalakan aktuator | Sistem menyalakan aktuator dan pada perangkat keras menyala | Sistem berhasil menyalakan aktuator dan pada perangkat keras menyala | Berhasil |
| Status aktuator | Sistem mampu menampilkan aktuator offline dan online | Berhasil menampilkan status sesuai dengan perangkat keras | Berhasil |

Selain itu terdapat pengujian otomatis yang telah berhasil dilakukan sesuai dengan perintah. Rincian dari pengujian dapat dilihat pada tabel 4.33.

Tabel 4.33 Rincian pengujian perangkat keras

| Sensor - aktuator | Case | Indikator Keberhasilan | Hasil |
|-------------------------------|---|---|----------|
| Sensor Debit – Selenoid Valve | Akan menyala sesuai dengan perintah secara otomatis dan mati sesuai dengan debit 5 ml | Selenoid Valve berhasil menyala dan mati sesuai kondisi | Berhasil |

4.1.3.4 Aktuator Cooling Pad

Cooling pad merupakan sistem untuk mengendalikan suhu dan kelembaban yang ada pada *greenhouse* yang dilakukan adalah dengan meneteskan air dari atas kebawah sehingga nantinya akan ditarik oleh *exhaust fan*, bentuk dari *cooling pad* berwarna coklat dengan bahan kardus, untuk jumlah dari *cooling pad* berjumlah 2 dan pompa yang berjumlah 2 untuk tempat kendali dari *cooling pad* adalah pompa yang terhubung dengan *mikrokontroller* sehingga berdasarkan kendali yang ada air akan menarik dari drum ke *cooling pad* dan air

akan turun dari atas hingga kebawah. Hasil dari pembuatan *cooling pad* dapat dilihat pada gambar.



Gambar 4.59 Implementasi alat Cooling Pad

Pengujian pertama adalah dengan melakukan input pada website apakah *website* dapat menyalakan *cooling pad*. Rincian dari pengujian dapat dilihat pada tabel 4.34.

Tabel 4.34 Pengujian website dan pengendalian *solenoid valve*

| No | Input pada Website | Kondisi Selenoid Cooling Pad | Keterangan |
|----|--------------------|------------------------------|------------|
| 1 | On | Nyala | Berhasil |
| 2 | Off | Mati | Berhasil |
| 3 | On | Nyala | Berhasil |
| 4 | Off | Mati | Berhasil |
| 5 | On | Nyala | Berhasil |
| 6 | Off | Mati | Berhasil |
| 7 | On | Nyala | Berhasil |
| 8 | Off | Mati | Berhasil |
| 9 | On | Nyala | Berhasil |
| 10 | Off | Mati | Berhasil |
| 11 | On | Nyala | Berhasil |
| 12 | Off | Mati | Berhasil |
| 13 | On | Nyala | Berhasil |
| 14 | Off | Mati | Berhasil |
| 15 | On | Nyala | Berhasil |
| 16 | Off | Mati | Berhasil |

Berdasarkan pengujian terdapat input dan juga hasil dari kondisi aktual *Cooling pad* telah sesuai sehingga respon dari *website* pada alat telah sesuai.

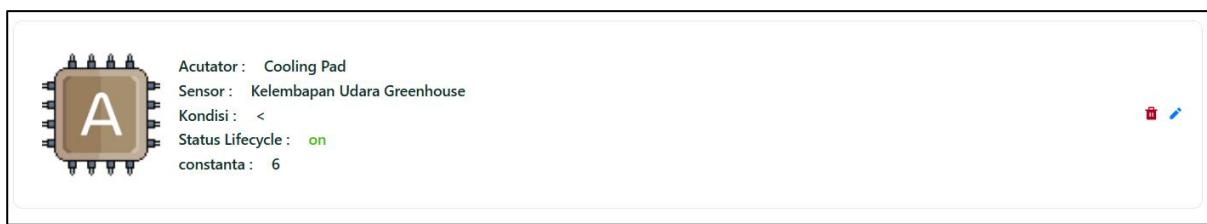
Pengujian kedua yang dilakukan adalah menguji *cooling pad* pada kondisi *offline* dan *online* sesuai dari *website*. Rincian dari pengujian yang dilakukan pada *cooling pad* dapat dilihat pada tabel 4.35.

Tabel 4.35 Pengujian cooling pad status actual cooling pad dan status website

| No | Status Alat Cooling Pad | Status Website | Keterangan |
|----|-------------------------|----------------|------------|
| 1 | Nyala | Online | Berhasil |
| 2 | Mati | Offline | Berhasil |
| 3 | Nyala | Online | Berhasil |
| 4 | Mati | Offline | Berhasil |
| 5 | Nyala | Online | Berhasil |
| 6 | Mati | Offline | Berhasil |
| 7 | Nyala | Online | Berhasil |
| 8 | Mati | Offline | Berhasil |
| 9 | Nyala | Online | Berhasil |
| 10 | Mati | Offline | Berhasil |

Berdasarkan pengujian terdapat kondisi aktual *cooling pad* dan kondisi *website* mengenai kondisi alat telah sesuai sehingga fitur *offline* dan *online* dari *website* pada *cooling pad* telah sesuai.

Pengujian terakhir adalah menguji otomatis yang dilakukan adalah menguji *cooling pad* dengan menginputkan otomatis berdasarkan sensor jika sensor menerima data kurang dari *range min* atau kelembaban 70% maka sistem *cooling pad* akan menyala lalu akan mati sesuai dari konstanta yakni di angka lebih dari 76% atau jika menerima data 76.1 akan mati. Rincian dari auatomatis dapat dilihat pada gambar 4.60.



Gambar 4.60 Rincian otomatis yang digunakan pada cooling pad

Berdasarkan perintah otomatis dilakukan pengujian. Rincian dari pengujian otomatis yang dilakukan pada *Cooling pad* dapat dilihat pada tabel 4.36.

Tabel 4.36 Rincian pengujian otomatis pada *Cooling pad*

| No. | Jam | Kondisi ideal | Nilai Kelembaban (persen) | Kondisi Cooling Pad | Keterangan |
|-----|----------|---------------|---------------------------|---------------------|------------|
| 1 | 21:10:10 | 70 – 80% | 72 % | Mati | Berhasil |
| 2 | 21:11:14 | 70 – 80% | 71 % | Mati | Berhasil |
| 3 | 21:12:11 | 70 – 80% | 69.5 % | Nyala | Berhasil |
| 4 | 21:13:20 | 70 – 80% | 70 % | Nyala | Berhasil |
| 5 | 21:14:27 | 70 – 80% | 72 % | Nyala | Berhasil |
| 6 | 21:15:31 | 70 – 80% | 76 % | Mati | Berhasil |
| 7 | 21:16:20 | 70 – 80% | 76.1 % | Mati | Berhasil |
| 8 | 21:17:12 | 70 – 80% | 69.5 % | Nyala | Berhasil |
| 9 | 21:18:23 | 70 – 80% | 76 % | Nyala | Berhasil |
| 10 | 21:10:18 | 70 – 80% | 76.1 % | Mati | Berhasil |

Berdasarkan pengujian *otomatis* yang telah dilakukan terjadi suhu berhasil menyala jika sudah kurang dari 70 % mulai dari 69.9% dan mati jika sesuai dari nilai kondisi minimum yakni 70 % ditambah dari konstanta yakni $70+6=76\%$, sehingga *cooling pad* akan mati jika kelembaban sudah berada di atas dari 76% yakni dimulai dari suhu 76%, berdasarkan pengujian otomatis menyala sesuai dengan kondisi sehingga dapat dikatakan *otomatis* dari *cooling pad* berdasarkan sensor telah berhasil.

Berdasarkan pengujian yang telah dilakukan dapat diketahui bahwa sistem dapat mematikan dan menyalakan aktuator, lalu dapat menampilkan status dari aktuator telah berhasil. Rincian dari pengujian yang dilakukan dapat dilihat pada tabel 4.37.

Tabel 4.37 Rincian pengujian black box dan alat *cooling pad*

| Fitur | Case | Indikator Keberhasilan | Hasil |
|--|---|--|----------|
| Sistem dapat mematikan dan menyalakan aktuator | Sistem menyalakan aktuator dan pada perangkat keras menyala | Sistem berhasil menyalakan aktuator dan pada perangkat keras menyala | Berhasil |
| Status aktuator | Sistem mampu menampilkan aktuator offline dan online | Berhasil menampilkan status sesuai dengan perangkat keras | Berhasil |

Selain itu terdapat pengujian otomatis yang telah berhasil dilakukan sesuai dengan perintah. Rincian dari pengujian dapat dilihat pada tabel 4.38.

Tabel 4.38 Rincian pengujian alat Cooling Pad

| Sensor - aktuator | Case | Indikator Keberhasilan | Hasil |
|--------------------------|--|------------------------------------|----------|
| Kelembaban - Cooling Pad | Akan menyala saat kelembaban kurang dari 70% | Cooling pad menyala sesuai kondisi | Berhasil |

4.1.3.5 Aktuator Pompa Nutrisi

Pompa Nutrisi berfungsi untuk mengalirkan nutrisi dari tandon untuk melon, pompa ini berfungsi untuk kendali pemberian nutrisi secara otomatis, pompa yang digunakan merupakan pompa ac pompa dihubungkan dengan pipa paralon berukuran 1 inch dengan menghubungkan antara tandon dengan pipa yang digunakan untuk memberikan nutrisi, pada bagian pompa terdapat kotak hitam yang berisi mengenai *mikrokontroller* sehingga pompa dapat dikendalikan arusnya untuk menyala dan mati berdasarkan kendali yang dilakukan *website*. Jumlah dari pompa yang ada adalah sebanyak 1 pompa, hasil dari pembuatan pompa nutrisi dapat dilihat pada gambar 4.61.



Gambar 4.61 Implementasi alat pompa nutrisi

Pengujian pertama adalah dengan melakukan input pada website apakah *website* dapat menyalakan pompa. Rincian dari pengujian dapat dilihat pada tabel 4.39.

Tabel 4.39. Pengujian pengendalian pompa dari website

| No | Input pada Website | Kondisi Pompa | Keterangan |
|----|--------------------|---------------|------------|
| 1 | On | Nyala | Berhasil |
| 2 | Off | Mati | Berhasil |

| No | Input pada Website | Kondisi Pompa | Keterangan |
|----|--------------------|---------------|------------|
| 3 | On | Nyala | Berhasil |
| 4 | Off | Mati | Berhasil |
| 5 | On | Nyala | Berhasil |
| 6 | Off | Mati | Berhasil |
| 7 | On | Nyala | Berhasil |
| 8 | Off | Mati | Berhasil |
| 9 | On | Nyala | Berhasil |
| 10 | Off | Mati | Berhasil |
| 11 | On | Nyala | Berhasil |
| 12 | Off | Mati | Berhasil |
| 13 | On | Nyala | Berhasil |
| 14 | Off | Mati | Berhasil |
| 15 | On | Nyala | Berhasil |
| 16 | Off | Mati | Berhasil |

Berdasarkan pengujian terdapat input dan juga hasil dari kondisi aktual dari pompa nutrisi telah sesuai sehingga respon dari *website* pada alat telah sesuai.

Pengujian kedua adalah menguji apakah kondisi aktual pompa dan juga website sudah sesuai. Rincian dari pengujian dapat dilihat pada tabel 4.40.

Tabel 4.40 Pengujian kondisi actual pompa dan website

| No | Status Pompa | Status Website | Keterangan |
|----|--------------|----------------|------------|
| 1 | Nyala | Online | Berhasil |
| 2 | Mati | Offline | Berhasil |
| 3 | Nyala | Online | Berhasil |
| 4 | Mati | Offline | Berhasil |
| 5 | Nyala | Online | Berhasil |
| 6 | Mati | Offline | Berhasil |
| 7 | Nyala | Online | Berhasil |
| 8 | Mati | Offline | Berhasil |

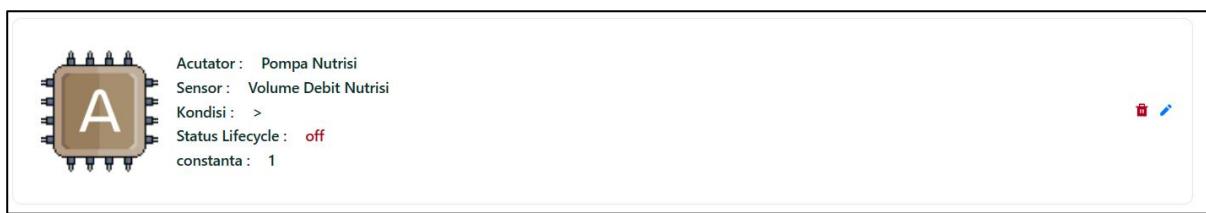
Berdasarkan pengujian terdapat kondisi aktual pompa nutrisi dan kondisi *website* mengenai kondisi alat telah sesuai sehingga fitur *offline* dan *online* dari *website* pada pompa nutrisi telah sesuai.

Pengujian terakhir adalah menguji otomatis yang dilakukan adalah menguji pompa dengan menginputkan otomatis dengan waktu mulai 19:50 dengan durasi menyala 1 menit dan perulangan dilakukan sebanyak 15 kali dimana interval antar alat menyala 1 menit. Tampilan input otomatis dapat dilihat pada gambar 4.62.



Gambar 4.62 Rincian input otomatis penjadwalan pada pompa

Selain pompa terdapat rincian dari otomatis berdasarkan sensor debit dimana jika kondisi melebihi range max dari sensor debit maka akan mati, setting untuk *range max* pada sensor debit adalah di angka 1000 untuk rincian dari otomatis dapat dilihat pada gambar 4.63.



Gambar 4.63 Rincian input otomatis berdasarkan sensor debit pada pompa

Berdasarkan perintah otomatis dilakukan pengujian. Rincian dari pengujian otomatis yang dilakukan pada pompa dapat dilihat pada tabel 4.41.

Tabel 4.41 Pengujian otomatis website pada pompa

| No | Waktu Nyala | Waktu Mati | Jumlah seharusnya (mililiter) | Jumlah sensor debit (mililiter) |
|----|-------------|------------|-------------------------------|---------------------------------|
| 1 | 19:50:02 | 19:50:11 | 1000 ml | 1473 ml |
| 2 | 19:52:04 | 19:52:14 | 1000 ml | 1443 ml |
| 3 | 19:54:01 | 19:54:10 | 1000 ml | 1543 ml |
| 4 | 19:56:02 | 19:56:11 | 1000 ml | 1653 ml |
| 5 | 19:58:02 | 19:58:11 | 1000 ml | 1753 ml |
| 6 | 20:00:01 | 20:00:10 | 1000 ml | 1543 ml |
| 7 | 20:03:02 | 20:03:11 | 1000 ml | 1563 ml |
| 8 | 20:06:01 | 20:06:11 | 1000 ml | 1893 ml |
| 9 | 20:08:02 | 20:08:12 | 1000 ml | 1593 ml |
| 10 | 20:10:01 | 20:10:12 | 1000 ml | 1763 ml |
| 11 | 20:12:02 | 20:12:10 | 1000 ml | 1433 ml |
| 12 | 20:14:01 | 20:14:11 | 1000 ml | 1593 ml |
| 13 | 20:16:02 | 20:16:11 | 1000 ml | 1453 ml |
| 14 | 20:18:01 | 20:18:10 | 1000 ml | 1533 ml |
| 15 | 20:20:01 | 20:20:10 | 1000 ml | 1573 ml |

Terjadi perbedaan yang cukup besar pada otomatis jumlah seharusnya dengan jumlah yang dikeluarkan oleh pompa nutrisi hal ini disebabkan pembacaan delay yang terlalu besar dari

sensor debit dan juga terdapat delay saat masuk ke server dan terakhir perintah dari otomatis berdasarkan sensor terdapat delay. Akan tetapi pada penjadwalan waktu delay dan penjadwalan di rentan 1 sampai 2 detik akan tetapi terdapat masalah jika setelah *pompa* dan *solenoid valve* mati tetapi air tetap mengalir.

4.1.3.6 Aktuator Exhaust Fan

Exhaust fan digunakan untuk menarik hawa panas yang ada pada *greenhouse* dan menarik air yang diteteskan oleh *cooling pad* sehingga adanya *exhaust fan* berfungsi untuk mengontrol suhu dari *greenhouse* sehingga suhu yang diberikan ideal, cara kerja dari *exhaust fan* adalah kipas menarik hawa panas yang ada dari dalam ke luar sehingga kipas menyala memberikan angin ke luar, jumlah dari *exhaust fan* adalah 4 buah dan untuk kendali dari internet terdapat box hitam yang berisi *mikrokontroller* sehingga system dapat menyalakan dan mematikan melalui *website*. Bentuk dari *Exhaust Fan* dapat dilihat pada gambar 4.64.



Gambar 4.64 Bentuk Aktuator *Exhaust Fan*

Pengujian pertama adalah dengan melakukan input pada website apakah *website* dapat menyalakan *Exhaust Fan*. Rincian dari pengujian dapat dilihat pada tabel 4.42.

Tabel 4.42 Pengujian respon pengendalian *exhaust fan* dari website

| No | Input pada Website | Kondisi Exhaust Fan | Keterangan |
|----|--------------------|---------------------|------------|
| 1 | On | Nyala | Berhasil |
| 2 | Off | Mati | Berhasil |
| 3 | On | Nyala | Berhasil |

| No | Input pada Website | Kondisi Exhaust Fan | Keterangan |
|----|--------------------|---------------------|------------|
| 4 | Off | Mati | Berhasil |
| 5 | On | Nyala | Berhasil |
| 6 | Off | Mati | Berhasil |
| 7 | On | Nyala | Berhasil |
| 8 | Off | Mati | Berhasil |
| 9 | On | Nyala | Berhasil |
| 10 | Off | Mati | Berhasil |

Berdasarkan pengujian terdapat input dan juga hasil dari kondisi aktual dari *exhaust fan* telah sesuai sehingga respon dari *website* pada alat telah sesuai.

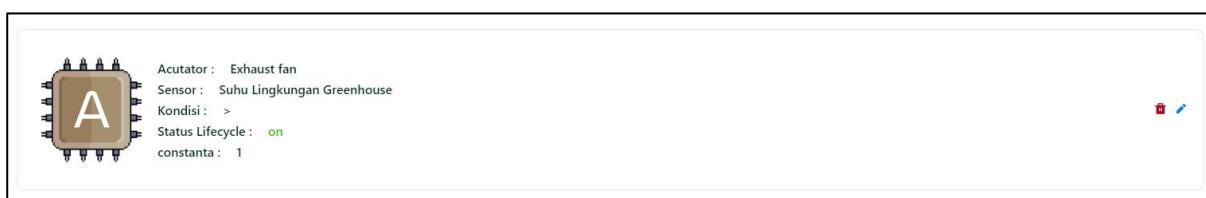
Pengujian kedua yang dilakukan adalah menguji *exhaust fan* pada kondisi *offline* dan *online* sesuai dari *website*. Rincian dari pengujian yang dilakukan pada *exhaust fan* dapat dilihat pada tabel 4.43.

Tabel 4.43 Pengujian respon website pada status actual exhaust fan

| No | Status Exhaust fan | Status Website | Keterangan |
|----|--------------------|----------------|------------|
| 1 | Nyala | Online | Berhasil |
| 2 | Mati | Offline | Berhasil |
| 3 | Nyala | Online | Berhasil |
| 4 | Mati | Offline | Berhasil |
| 5 | Nyala | Online | Berhasil |
| 6 | Mati | Offline | Berhasil |
| 7 | Nyala | Online | Berhasil |
| 8 | Mati | Offline | Berhasil |
| 9 | Nyala | Online | Berhasil |
| 10 | Mati | Offline | Berhasil |

Berdasarkan pengujian terdapat kondisi aktual *exhaust fan* dan kondisi *website* mengenai kondisi alat telah sesuai sehingga fitur *offline* dan *online* dari *website* pada *exhaust fan* telah sesuai.

Pengujian terakhir adalah menguji otomatis yang dilakukan adalah menguji *exhaust fan* dengan menginputkan otomatis berdasarkan sensor jika sensor melebihi dari range max atau 30°C dari sensor suhu lingkungan dilakukan status menyalakan alat dan akan mati sesuai dengan range max dikurang konstanta sehingga jika suhu kurang dari 29°C maka *exhaust fan* akan mati. Rincian dari auatomatis dapat dilihat pada gambar 4.66.



Gambar 4.66 Rincian otomatis actuator exhaust fan berdasarkan sensor suhu

Berdasarkan perintah otomatis dilakukan pengujian. Rincian dari pengujian otomatis yang dilakukan pada *exhaust fan* dapat dilihat pada tabel 4.44.

Tabel 4.44 Pengujian otomatis exhaust fan

| No | Jam | Nilai Suhu (Celcius) | Kondisi Kipas | Keterangan |
|----|----------|----------------------|---------------|------------|
| 1 | 16:35:16 | 31 °C | Nyala | Berhasil |
| 2 | 16:38:40 | 28 °C | Mati | Berhasil |
| 3 | 16:40:15 | 30.5 °C | Nyala | Berhasil |
| 4 | 16:43:20 | 28.9 °C | Mati | Berhasil |
| 5 | 09:16:27 | 30 °C | Mati | Berhasil |
| 6 | 09:16:31 | 30.1 °C | Nyala | Berhasil |
| 7 | 09:22:20 | 29.6 °C | Nyala | Berhasil |
| 8 | 09:26:12 | 28.9 °C | Mati | Berhasil |
| 9 | 09:27:23 | 29.2 °C | Mati | Berhasil |
| 10 | 10:13:18 | 30.2°C | Nyala | Berhasil |

Berdasarkan pengujian otomatis yang telah dilakukan terjadi suhu berhasil menyala jika sudah melebih kondisi 30 °C mulai dari 30.1°C derajat dan mati jika sesuai dari nilai kondisi maksimum yakni 30°C dikurang dari konstanta yakni $30-1=29$, sehingga *cooling pad* akan mati jika suhu sudah berada dibawah dari 29 °C yakni dimulai dari suhu 28.9 °C, berdasarkan pengujian otomatis menyala sesuai dengan kondisi sehingga dapat dikatakan *otomatis* berdasarkan sensor telah berhasil.

Berdasarkan pengujian yang telah dilakukan dapat diketahui bahwa sistem dapat mematikan dan menyalakan aktuator, lalu dapat menampilkan status dari aktuator telah berhasil. Rincian dari pengujian yang dilakukan dapat dilihat pada tabel 4.45.

Tabel 4.45 Rincian pengujian black box website pada exhaust fan

| Fitur | Case | Indikator Keberhasilan | Hasil |
|--|---|--|----------|
| Sistem dapat mematikan dan menyalakan aktuator | Sistem menyalakan aktuator dan pada perangkat keras menyala | Sistem berhasil menyalakan aktuator dan pada perangkat keras menyala | Berhasil |
| Status aktuator | Sistem mampu menampilkan aktuator offline dan online | Berhasil menampilkan status sesuai dengan perangkat keras | Berhasil |

Selain itu terdapat pengujian otomatis yang telah berhasil dilakukan sesuai dengan perintah. Rincian dari pengujian dapat dilihat pada tabel 4.46.

Tabel 4.46 Rincian pengujian alat exhaust fan

| Fitur | Case | Indikator Keberhasilan | Hasil |
|-------|------|------------------------|-------|
| | | | |

| | | | |
|--------------|--|---|----------|
| Suhu – Kipas | Akan menyala jika suhu mencapai 30 derajat Celcius dan mati pada suhu 28 derajat celcius | Kipas dan Cooling Pad Menyala dan mati sesuai kondisi | Berhasil |
|--------------|--|---|----------|

4.2 Deploy

Pada bagian *deploy* adalah agar *website* dapat diakses oleh banyak orang hasil dari *hosting* adalah *link* dari *api* dan juga *website*. Hasil dari *deploy website* adalah www.iterahero.com dan juga hasil *hosting* dari *backend* adalah <https://iterahero.fly.dev> . Link tersebut dapat diakses oleh semua orang dapat mencoba ataupun melakukan testing pada sub bab 4.3 yakni pengujian sus.

4.3 Pengujian SUS

Pada pengujian SUS telah dilakukan dengan jumlah responden 20 orang, dengan 1 orang pemilik *greenhouse* yakni ibu Zunanik Mufidah, S.TP., M.Si, 2 orang penjaga *greenhouse*, 1 orang konsultan *greenhouse*, 1 orang IT, dan juga 15 mahasiswa. Rincian dari pengujian sus yang telah dilakukan dapat dilihat pada tabel 4.47.

Tabel 4.47 Perhitungan SUS

| Responden | Pertanyaan | | | | | | | | | | jumlah | Jumlah * 2.5 |
|--------------|------------|---|---|---|---|---|---|---|---|----|--------|--------------|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | | |
| Responden 1 | 5 | 1 | 2 | 2 | 5 | 1 | 5 | 1 | 5 | 2 | 35 | 87.5 |
| Responden 2 | 5 | 4 | 4 | 4 | 4 | 1 | 3 | 3 | 2 | 3 | 23 | 57.5 |
| Responden 3 | 5 | 2 | 5 | 2 | 5 | 1 | 4 | 1 | 4 | 1 | 36 | 90 |
| Responden 4 | 4 | 2 | 3 | 3 | 4 | 1 | 5 | 1 | 2 | 3 | 28 | 70 |
| Responden 5 | 5 | 1 | 1 | 1 | 5 | 5 | 5 | 4 | 5 | 4 | 26 | 65 |
| Responden 6 | 5 | 2 | 3 | 3 | 5 | 1 | 3 | 1 | 4 | 2 | 31 | 77.5 |
| Responden 7 | 5 | 1 | 1 | 1 | 5 | 2 | 5 | 1 | 1 | 1 | 31 | 77.5 |
| Responden 8 | 5 | 1 | 1 | 1 | 5 | 1 | 5 | 1 | 5 | 2 | 35 | 87.5 |
| Responden 9 | 4 | 1 | 3 | 2 | 5 | 2 | 5 | 2 | 4 | 3 | 31 | 77.5 |
| Responden 10 | 5 | 2 | 5 | 2 | 5 | 2 | 5 | 1 | 5 | 2 | 36 | 90 |
| Responden 11 | 3 | 2 | 4 | 4 | 4 | 2 | 4 | 2 | 4 | 4 | 25 | 62.5 |
| Responden 12 | 4 | 2 | 5 | 3 | 4 | 2 | 5 | 5 | 5 | 3 | 28 | 70 |
| Responden 13 | 4 | 1 | 4 | 1 | 5 | 3 | 3 | 2 | 4 | 4 | 29 | 72.5 |

| Responden | Pertanyaan | | | | | | | | | | jumlah | Jumlah * 2.5 |
|--------------|------------|---|---|---|---|---|---|---|---|----|--------|--------------|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | | |
| Responden 14 | 4 | 3 | 4 | 2 | 4 | 3 | 3 | 2 | 4 | 4 | 25 | 62.5 |
| Responden 15 | 5 | 2 | 5 | 2 | 5 | 1 | 4 | 2 | 4 | 1 | 35 | 87.5 |
| Responden 16 | 5 | 5 | 5 | 4 | 4 | 2 | 4 | 1 | 5 | 2 | 29 | 72.5 |
| Responden 17 | 5 | 1 | 5 | 5 | 5 | 1 | 5 | 1 | 5 | 2 | 35 | 87.5 |
| Responden 18 | 4 | 4 | 4 | 3 | 5 | 4 | 5 | 4 | 4 | 5 | 22 | 55 |
| Responden 19 | 5 | 1 | 5 | 5 | 5 | 5 | 4 | 1 | 5 | 5 | 27 | 67.5 |
| Responden 20 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 40 | 100 |
| Responden 21 | 4 | 2 | 5 | 2 | 5 | 2 | 4 | 2 | 4 | 4 | 30 | 75 |
| Rata-rata | | | | | | | | | | | | 75.83 |

Berdasarkan perhitungan yang telah dilakukan dapat disimpulkan bahwa nilai dari pengujian sus adalah 75.875 yang memiliki arti *good* atau baik dan dapat diterima oleh user sehingga kesimpulan dari pengujian sus ini sistem dapat diterima oleh user. Pada bagian yang kurang dengan hasil rata-rata ragu-ragu terletak pada masalah pertanyaan ke 3 mengenai sistem mudah untuk digunakan, lalu pertanyaan 4 mengenai Saya membutuhkan bantuan dari orang lain atau teknisi dalam menggunakan sistem ini dan pertanyaan 10 mengenai pengguna perlu membiasakan sistem. Berdasarkan kesimpulan tersebut sistem perlu adanya *guidebook* dan juga pengguna perlu diajari terlebih dahulu untuk menggunakan sistem.

4.4 Launch

Pada bagian launch sistem akan digunakan dalam *greenhouse* untuk melakukan monitoring dari *greenhouse*. Penanaman melon dilakukan dengan sistem yang sudah dibuat untuk dilakukan perbandingan nantinya. Permbuatan alat berhasil dilakukan terdapat data-data yang didapat mengenai kondisi *greenhouse* setelah adanya sistem ini.

4.4.1 Kondisi Suhu

Setelah teknologi untuk melakukan monitoring dilakukan pengecekan pada suhu yang ada pada *greenhouse*. Rincian dari suhu setelah adanya *exhaust fan* pada siang hari dengan kondisi cerah pada *greenhouse* dapat dilihat pada tabel 4.48.

Tabel 4.48 Kondisi suhu *greenhouse* setelah dilakukan pemasangan IoT

| No | Jam | Kondisi ideal (celcius) | Nilai Suhu (celcius) | Suhu di luar Greenhouse (celcius) |
|----|-----------|-------------------------|----------------------|-----------------------------------|
| 1 | 09:02 wib | 22-30 °C | 28.04 °C | 28 °C |

| | | | | |
|----|-----------|----------|----------|-------|
| 2 | 10:04 wib | 22-30 °C | 31.91 °C | 34 °C |
| 3 | 11:07 wib | 22-30 °C | 34.24 °C | 39 °C |
| 4 | 12:04 wib | 22-30 °C | 33.99 °C | 39 °C |
| 5 | 13:31 wib | 22-30 °C | 33.28 °C | 37 °C |
| 6 | 14:22 wib | 22-30 °C | 32.8 °C | 34 °C |
| 7 | 15:03 wib | 22-30 °C | 31.29 °C | 37 °C |
| 8 | 16:32 wib | 22-30 °C | 28.90 °C | 29 °C |
| 9 | 17:10 wib | 22-30 °C | 29.4 °C | 28 °C |
| 10 | 18:01 wib | 22-30 °C | 28.73 °C | 27 °C |

Dengan adanya sistem monitoring didapatkan data suhu pada *greenhouse* dan terdapat perbedaan antara menggunakan *exhaust fan* dan tidak perbedaan terbanyak adalah 6 derajat celcius, akan tetapi suhu masih belum efektif pada *greenhouse* saat di siang hari dan cuaca cerah yakni pada sian hari mencapai kondisi ideal yakni antara 22-30 derajat celcius salah satu masalahnya adalah panjang dari *greenhouse* dan juga kekuatan dari putaran kipas sehingga pada penelitian lanjutan sebaiknya dapat melakukan riset mengenai efektivitas dari *exhaust fan* dan *cooling pad* sehingga kondisi *greenhouse* dapat lebih stabil

4.4.2 Kondisi Kelembaban

Pemasangan sistem dari IoT dan juga *website* didapatkan kondisi dari kelembaban yang ada pada siang hari terdapat hasil dari pembacaan sensor dht pada tingkat kelembaban yang ada pada *greenhouse*. Rincian dari kendali *cooling pad* pada kelembaban dapat dilihat pada tabel 4.49.

Tabel 4.49 Kondisi kelembaban *greenhouse* setelah dilakukan pemasangan IoT

| No | Jam | Kondisi ideal (persen) | Nilai Kelembaban (persen) |
|----|-----------|------------------------|---------------------------|
| 1 | 09:10 wib | 70-80% | 71.38 % |
| 2 | 10:01 wib | 70-80% | 72.48 % |
| 3 | 11:03 wib | 70-80% | 64.66 % |
| 4 | 12:04 wib | 70-80% | 66.77 % |
| 5 | 13:10 wib | 70-80% | 67.13 % |
| 6 | 14:05 wib | 70-80% | 73.78 % |
| 7 | 15:04 wib | 70-80% | 75.48 % |
| 8 | 16:07 wib | 70-80% | 76.21 % |
| 9 | 17:13 wib | 70-80% | 77.85 % |
| 10 | 18:09 wib | 70-80% | 80.1 % |

Kondisi kelembaban yang ada pada *greenhouse* terjadi penurunan yang drastis disiang hari dikarenakan *cooling pad* pada siang hari dapat terjadi kekosongan hal ini dikarenakan drum yang berisi air dapat habis karena kekeringan oleh suhu yang panas pada siang hari

sehingga jika air pada drum telah habis drum perlu di isi secara manual sehingga *cooling pad* akan menyala dan meneteskan air agar dapat berfungsi semestinya. Akan tetapi efektifitas pada *cooling pad* cukup baik untuk menciptakan kondisi kelembaban yang ideal.

4.4.3 Kondisi Pemberian Nutrisi

Kondisi pemberian nutrisi didasarkan oleh sensor debit, *solenoid valve*, dan *pompa air* yang akan mengaliri nutrisi sesuai dengan waktu dan mati berdasarkan pembacaan sensor debit di 6 liter. Rincian dari kondisi pemberian nutrisi dapat dilihat pada tabel 4.50.

Tabel 4.50 Kondisi pemberian nutrisi pada *greenhouse* setelah dilakukan pemasangan IoT

| No | Jam | Kondisi ideal (ml) | Nilai Debit (ml) | Keterangan |
|----|-----------|--------------------|------------------|------------|
| 1 | 08:02 wib | 6000 ml | 6190 ml | Berhasil |
| 2 | 10:03 wib | 6000 ml | 6540 ml | Berhasil |
| 3 | 12:04 wib | 6000 ml | 6760 ml | Berhasil |
| 4 | 14:05 wib | 6000 ml | 6134 ml | Berhasil |
| 5 | 16:00 wib | 6000 ml | 4440 ml | Gagal |

Pada *greenhouse* melon ITERA menghasilkan data-data yang cukup mirip antara nilai debit dan kondisi ideal dari pemberian nutrisi akan tetapi terdapat masalah pada nilai debit yang terkadang tidak dilakukan penyiraman seperti pada pemberian nomor 5. Hal ini dikarenakan masalah pada pemberian nutrisi saat tandon habis sehingga alat menyala akan tetapi tidak mengeluarkan nutrisi sehingga hal ini dapat berdampak pada hasil dari penanaman melon. Sehingga kondisi ini diperlukannya penjaga *greenhouse* untuk mengecek tandon, atau dapat memperkirakan besar dari tandon. Dan untuk penelitian selanjutnya dapat dilakukan pembuatan sistem otomatis pengisian tandon dikarenakan dalam peracikan cukup memakan waktu, atau jika tandon habis perlu adanya pengingat untuk pengguna mengisi tandon kembali untuk mencegah penyiraman yang gagal.

4.4.1 Hasil penanaman melon

Setelah dilakukan penanaman melon terjadi perbedaan antara penanaman melon menggunakan sistem konvensional dan sistem *greenhouse* yang menggunakan metode IoT. Sistem konvensional yang dilakukan adalah dengan cara menyiram nutrisi secara manual dengan menggunakan gelas ukuran dari gelas dapat dilihat pada gambar 4.65.



Gambar 4.65 Ukuran gelas yang digunakan untuk metode konvensional

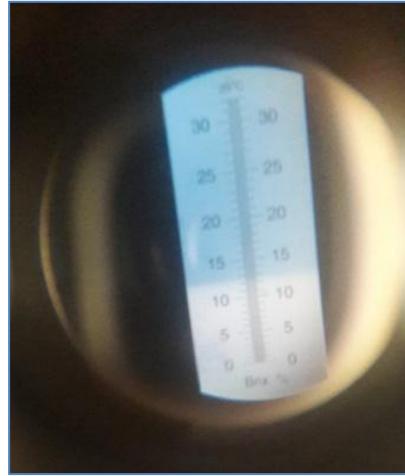
Perbedaan dilakukan mulai dari berat melon menggunakan timbangan, lalu diameter melon menggunakan meteran, dan terakhir adalah tingkat kemanisan melon menggunakan *refractometer* untuk satuan ukur dari tingkat kemanisan dengan satuan brix yang merupakan derajat satuan untuk menggambarkan jumlah atau kadar kandungan gula (zat padat) yang terlarut dalam larutan air. Gambar dari *refactometer* dapat dilihat pada gambar 4.66.



Gambar 4.66 *refractometer* alat ukur tingkat kemanisan melon

Dilakukan penimbangan, pengukuran diameter, dan juga pengukuran tingkat kemanisan dari melon gambaran dari perbedaan penanaman melon dapat dilihat pada tabel 4.51.

Tabel 4.51 Gambaran perbedaan penanaman melon menggunakan metode konvensional dan *smart system*

| Melon dengan penanaman konvensional | Melon penanaman <i>smart system</i> |
|--|---|
|  |  |
|  |  |

Berdasarkan gambar diatas didapatkan hasil melon metode konvensional lebih kecil dibandingkan melon pada metode *smart system*. Detail dari perbedaan metode penanaman melon dapat dilihat pada tabel 4.52.

Tabel 4.52 Perbedaan sistem konvensional dan *Smart Greenhouse*

| | Parameter | Melon <i>Smart Greenhouse</i> | Melon konvensional |
|---|-------------------------|-------------------------------|--------------------|
| 1 | Berat Melon | 800gram – 1,6 kg | 800 gram – 1 kg |
| 2 | Diameter Melon | 8cm – 11 cm | 6 cm – 8cm |
| 3 | Tingkat kemanisan melon | +- 12 brix | +- 14 brix |

Berdasarkan data yang didapatkan pada tabel 4.51 berat dan diameter melon pada sistem yang menggunakan *smart greenhouse* memiliki ukuran yang lebih besar dibandingkan dengan diameter melon yang menggunakan sistem konvensional. Perbedaan terjadi dari pemberian nutrisi menggunakan irigasi tetes sedangkan manual menyiram dengan menggunakan gelas sehingga terjadi perbedaan dari Teknik penyiraman dan juga pemberian

nutrisi yang memiliki perbedaan pada jumlah pemberian nutrisi, nutrisi diberikan lebih banyak pada metode *konvensional*.

Terjadi permasalan tingkat kemanisan melon pada *smart greenhouse* hal ini dikarenakan terjadi permasalahan pada pemberian nutrisi yang dilakukan jika tandon habis dan juga terdapat permasalahan pada berat melon yang beragam hal ini dapat dikarenakan oleh beberapa faktor, faktor yang menyebabkanya adalah irigasi tetes yang dilakukan tersumbat atau tidak keluar sesuai dengan yang dibutuhkan, terdapat salah satu factor yang menjadikan melon kecil adalah jumlah buah pada 1 tanaman dan juga jumlah cabang dan daun yang terlalu banyak sehingga nutrisi dapat berpindah ke tingkat daun ataupun pada cabang batang lainnya sehingga menyebabkan nutrisi yang dibutuhkan buah untuk manis menjadi kurang. Masalah lainnya yang terjadi adalah tidak adanya pengendalian matahari pada *greenhouse* salah satu faktor tingkat kemanisan dan besar buah adalah sinar matahari sehingga sistem selanjutnya diharapkan dapat meningkatkan tingkat kemanisan dari buah.

BAB V

KESIMPULAN

5.1 Kesimpulan

Berdasarkan pengujian yang telah dilakukan pada sistem monitoring *smart greenhouse* berbasis IoT dapat disimpulkan sebagai berikut :

1. Metode agile kanban dapat digunakan sebagai metode untuk mengembangkan sistem berbasis IoT. Metode ini dimulai dari merencanakan sistem yang dibuat *user story* dan kebutuhan fungsional lalu ke tahap pembuatan desain sistem seperti diagram uml dan untuk implementasi dengan menyelesaikan seluruh story yang ada dalam *board* dilanjutkan dengan testing, deploy, dan pengujian pada pengguna, jika pengujian sudah baik maka dilanjutkan pada penggunaan sistem.
2. Hasil dari evaluasi sistem monitoring *smart greenhouse* berbasis IoT dengan menggunakan metode *System Usability Scale* (SUS) memiliki skor di 75.875 sistem dapat diterima oleh user dengan baik. Selain itu pengujian dilakukan dengan metode black box testing yang telah dilakukan berhasil melewati semua case yang ada sehingga sistem berhasil lulus dari pengujian black box.
3. Pada tanaman melon yang menggunakan sistem berbasis IoT memiliki diameter buah yang lebih besar dan lebih berat akan tetapi pada tingkat kemanisan melon konvensional lebih manis hal ini dikarenakan masih kurangnya nutrisi yang diberikan pada tanaman melon berbasis IoT.

5.2 Saran

Setelah melaksanakan penelitian sistem monitoring *smart greenhouse* berbasis IoT. Adapun saran yang dapat diberikan agar sistem monitoring *smart greenhouse* dapat bejalan lebih baik :

1. Perlu adanya riset lebih lanjut mengenai keluarnya debit pada irigasi tetes.
2. Perlu adanya riset efektifitas pada pendigangan *greenhouse*.
3. Sistem website adanya penambahan aplikasi sehingga dapat dimonitoring tanpa harus membuka *browser* terlebih dahulu.
4. Perlunya sistem peracikan otomatis pada tandon ataupun melakukan monitoring pada tandon jika tandon sudah habis nutrisi.
5. Adanya sistem yang dapat mengontrol sinar matahari pada *greenhouse*

DAFTAR PUSTAKA

- [1] "Kebun Raya ITERA, Pusat Konservasi Tanaman Sumatera - ITERA." <https://www.itera.ac.id/kebun-raya-itera-pusat-konservasi-tanaman-sumatera/> (accessed Jan. 26, 2023).
- [2] W. Wahyudi, E. Andriani, and A. Nurmelia, "PENDAPATAN DAN STRATEGI PEMASARAN PETANI MELON DI KABUPATEN SELUMA," *Agritepa: J. Ilm. dan Tek. Per.*, vol. 7, no. 1, pp. 57–69, May 2020, doi: [10.3767/agritepa.v7i1.999](https://doi.org/10.3767/agritepa.v7i1.999).
- [3] N. N. Lidyawati and I. N. Suwastika, "Perbanyakkan Tanaman Melon (*Cucumis melo* L.) Secara In Vitro Pada Medium Ms Dengan Penambahan Indole Acetic Acid (IAA) Dan Benzil Amino Purin (BAP)," vol. 1, 2012.
- [4] "ALISHA F1." <https://www.panahmerah.id/product/alisha> (accessed Mar. 26, 2023).
- [5] I. K. A. Indrawan, I. G. A. Gunadi, and I. W. Wiraatmaja, "Pengaruh Jenis Media Tanam dan Varietas terhadap Hasil Tanaman Melon (*Cucumis melo* L.) pada Sistem Irigasi Tetes," vol. 10, no. 3, 2021.
- [6] Departemen Biologi, Fakultas Matematika dan Ilmu Pengetahuan Alam, Institut Pertanian Bogor, Kampus IPB Darmaga, Bogor 16680, T. Triadiati, M. Muttaqin, Departemen Biologi, Fakultas Matematika dan Ilmu Pengetahuan Alam, Institut Pertanian Bogor, Kampus IPB Darmaga, Bogor 16680, N. Saidah Amalia, and Departemen Biologi, Fakultas Matematika dan Ilmu Pengetahuan Alam, Institut Pertanian Bogor, Kampus IPB Darmaga, Bogor 16680, "Growth, Yield, and Fruit of Melon Quality Using Silica Fertilizer," *JIPI*, vol. 24, no. 4, pp. 366–374, Oct. 2019, doi: [10.18343/jipi.24.4.366](https://doi.org/10.18343/jipi.24.4.366).
- [7] Sudaryono, Sudaryono. "Pengaruh Naungan Dan Pemberian Mulsa Terhadap Produksi Buah Melon (*Cucumis Melo* L.) (Studi Kasus Di Pantai Bugel, Kabupaten Kulon Progo)." *Jurnal Teknologi Lingkungan BPPT*, vol. 6, no. 3, 2005, doi:10.29122/jtl.v6i3.353.
- [8] F. Susanto, N. K. Prasiani, and P. Darmawan, "IMPLEMENTASI INTERNET OF THINGS DALAM KEHIDUPAN SEHARI-HARI," *imagine*, vol. 2, no. 1, pp. 35–40, Apr. 2022, doi: [10.35886/imagine.v2i1.329](https://doi.org/10.35886/imagine.v2i1.329).
- [9] Y. Efendi, "INTERNET OF THINGS (IOT) SISTEM PENGENDALIAN LAMPU MENGGUNAKAN RASPBERRY PI BERBASIS MOBILE," vol. 4, no. 1, 2018.
- [10] K. U. Ariawan, "PENERAPAN IoT UNTUK SISTEM KENDALI JARAK JAUH PERALATAN LISTRIK RUMAH TANGGA BERBASIS RASPBERRY PI," *j. nas.*

pendidik. teknik. inform., vol. 9, no. 3, p. 292, Dec. 2020, doi: [10.23887/janapati.v9i3.23264](https://doi.org/10.23887/janapati.v9i3.23264).

- [11] I. Erlangga Prasetya, S. Achmadi, and D. Rudhistiar, “PENERAPAN IOT (INTERNET OF THINGS) UNTUK SISTEM MONITORING AIR DAN CONTROLLING PADA KOLAM IKAN GURAMI BERBASIS WEBSITE,” *jati*, vol. 6, no. 2, pp. 1184–1191, Jan. 2023, doi: [10.36040/jati.v6i2.5400](https://doi.org/10.36040/jati.v6i2.5400).
- [12] Hardani, Dian & Hayat, Latiful. (2020). Penerapan Internet of Things (IoT) pada Sistem Pengendali dan Pengaman Pintu Berbasis Android. *Jurnal Riset Rekayasa Elektro*. 2. 10.30595/jrre.v2i2.9056.
- [13] “Perbedaan Mobile Aplikasi Dan Mobile Web|S1 Sistem Komputer S.Kom.” <http://sistem-komputer-s1.stekom.ac.id/informasi/baca/Perbedaan-Mobile-Aplikasi-Dan-Mobile-Web/58b4f7f452cd95b4715b3436b25a14ac778530e6> (accessed Mar. 27, 2023).
- [14] Gurung, Gagan & Shah, Rahul & Jaiswal, Dhiraj. (2020). Software Development Life Cycle Models-A Comparative Study. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*. 30-37. 10.32628/CSEIT206410.
- [15] “Apa Itu Agile? Pengertian, Prinsip, Metode, dan Kelebihan [Terlengkap].” <https://www.niagahoster.co.id/blog/agile-adalah/> (accessed Apr. 13, 2023).
- [16] “Kanban - A brief introduction | Atlassian.” <https://www.atlassian.com/agile/kanban> (accessed Apr. 13, 2023).
- [17] P. Suparman and M. Huda, “PENERAPAN KANBAN AGILE DEVELOPMENT DALAM PENGEMBANGAN SISTEM MANAJEMEN SKRIPSI DAN TUGAS AKHIR STMIK CIKARANG MENGGUNAKAN FRAMEWORK CODEIGNITER,” 2021.
- [18] “IoT Connectivity: Choosing a Black-Box, White-Box, or Gray-Box Approach - Embedded Computing Design.” <https://embeddedcomputing.com/technology/IoT/IoT-connectivity-choosing-a-black-box-white-box-or-gray-box-approach> (accessed Apr. 14, 2023).
- [19] Brooke, John. (2013). SUS: a retrospective. *Journal of Usability Studies*. 8. 29-40.
- [20] A. Sidik, “Penggunaan System Usability Scale (SUS) Sebagai Evaluasi Website Berita Mobile,” *Technologia*, vol. 9, no. 2, p. 83, Apr. 2018, doi: [10.31602/tji.v9i2.1371](https://doi.org/10.31602/tji.v9i2.1371).

- [21] A. P. Merdekawan and P. Sari, “Studi Awal Rancang Bangun Indoor Farming Monitoring System Berbasis IoT dengan Protokol Websocket,” *JIKA*, vol. 9, no. 2, pp. 189–198, Nov. 2022, doi: 10.29244/jika.9.2.189-198.
- [22] A. Prihanto, N. Rachmawati, and A. Prapanca, “Smart Garden Automation Dengan Memanfaatkan Teknologi Berbasis Internet Of Things (IoT),” *JIEET*, vol. 5, no. 2, pp. 55–60, Dec. 2021, doi: [10.26740/jieet.v5n2.p55-60](https://doi.org/10.26740/jieet.v5n2.p55-60).
- [23] M. S. Farooq, R. Javid, S. Riaz, and Z. Atal, “IoT Based Smart *Greenhouse* Framework and Control Strategies for Sustainable Agriculture,” *IEEE Access*, vol. 10, pp. 99394–99420, 2022, doi: [10.1109/ACCESS.2022.3204066](https://doi.org/10.1109/ACCESS.2022.3204066).
- [24] B. Lanitha *et al.*, “IoT Enabled Sustainable Automated *Greenhouse* Architecture with Machine Learning Module,” *Journal of Nanomaterials*, vol. 2022, pp. 1–6, Jun. 2022, doi: [10.1155/2022/1314903](https://doi.org/10.1155/2022/1314903).
- [25] M. Y. Muhammin, A. Rahma Annisa, and B. Montolalu, “Rancang Bangun Smart System Green House untuk Budidaya Melon Berbasis PLC,” *J. Technol. Inform.*, vol. 4, no. 1, pp. 26–30, Oct. 2022, doi: [10.37802/joti.v4i1.260](https://doi.org/10.37802/joti.v4i1.260).
- [26] R. S. Amrullah, “PENGEMBANGAN SISTEM MONITORING KEGIATAN BELAJAR MENGAJAR DAN MEDIA PEMBELAJARAN SHOLAT,” vol. 1, no. 2, 2017.
- [27] S. Pamungkas, “Smart *Greenhouse* System On Paprican Plants Based On Internet of Things,” *Jurnal Ilmiah Telekomunikasi, Kendali dan Elektronika Terapan*, vol. 7, no. 2, pp. 197–207, Feb. 2020, doi: [10.34010/telekontran.v7i2.2277](https://doi.org/10.34010/telekontran.v7i2.2277).
- [28] R. Faizin, D. Darmansyah, and Y. Darnila, “PERENCANAAN MODEL BISNIS *GREENHOUSE* DI KABUPATEN ACEH TENGAH MENGGUNAKAN METODE BUSINESS MODEL CANVAS”.
- [29] Q. Syadza and A. G. Permana, “PENGONTROLAN DAN MONITORING PROTOTYPE GREEN HOUSE MENGGUNAKAN MIKROKONTROLER DAN FIREBASE”.
- [30] F. Susanto, N. K. Prasiani, and P. Darmawan, “IMPLEMENTASI INTERNET OF THINGS DALAM KEHIDUPAN SEHARI-HARI,” *imagine*, vol. 2, no. 1, pp. 35–40, Apr. 2022, doi: [10.35886/imagine.v2i1.329](https://doi.org/10.35886/imagine.v2i1.329).
- [31] S. N. Simbolon, “Pengaruh Interval Waktu Pemberian Nutrisi Ab-Mix dan Metode Hidroponik pada Tanaman Melon (*Cucumis melo L.*),” vol. 6, 2018.
- [32] A. Prihanto, N. Rachmawati, and A. Prapanca, “Smart Garden Automation Dengan

- Memanfaatkan Teknologi Berbasis Internet Of Things (IoT)," *JIEET*, vol. 5, no. 2, pp. 55–60, Dec. 2021, doi: [10.26740/jieet.v5n2.p55-60](https://doi.org/10.26740/jieet.v5n2.p55-60).
- [33] M. S. Son, "PENGEMBANGAN MIKROKONTROLER SEBAGAI REMOTE CONTROL BERBASIS ANDROID," *J. Teknik Informatika*, vol. 11, no. 1, pp. 67–74, May 2018, doi: [10.15408/jti.v11i1.6293](https://doi.org/10.15408/jti.v11i1.6293).
- [34] N. S. Abu *et al.*, "Internet of Things Applications in Precision Agriculture: A Review," *Journal of Robotics and Control*, vol. 3, no. 3, pp. 338–347, May 2022, doi: [10.18196/jrc.v3i3.14159](https://doi.org/10.18196/jrc.v3i3.14159).
- [35] M. S. Anshori, S. R. Akbar, and R. Maulana, "Implementasi Sistem Sensor Dan Aktuator Real Time Pada Tanaman Jamur".
- [36] A. G. Gani, "PENGENALAN TEKNOLOGI INTERNET SERTA DAMPAKNYA".
- [37] N. P. Windryani, "ANALISA PERBANDINGAN PROTOKOL MQTT DENGAN HTTP PADA IOT PLATFORM PATRIOT".
- [38] D. W. L. Pamungkas and S. Rochimah, "Pengujian Aplikasi Web - Tinjauan Pustaka Sistematis," *J. IPTEK*, vol. 23, no. 1, pp. 17–24, 2019, doi: [10.31284/j.iptek.2019.v23i1.459](https://doi.org/10.31284/j.iptek.2019.v23i1.459).
- [39] C. Frederick and S. Bernard, "Analisa dan Desain Sistem Bimbingan Tugas Akhir Berbasis Web dengan Studi Kasus Fakultas Teknologi Informasi," *J. Inform.*, vol. 1, no. 2, pp. 93–106, 2005.
- [40] O. Pahlevi, A. Mulyani, and M. Khoir, "SISTEM INFORMASI INVENTORY BARANG MENGGUNAKAN METODE OBJECT ORIENTED DI PT . LIVAZA TEKNOLOGI INDONESIA JAKARTA," vol. 5, no. 1, 2018.
- [41] S. M. Prasetiyo, M. I. P. Nugroho, R. L. Putri, and O. Fauzi, "Pembahasan Mengenai Front-End Web Developer dalam Ruang Lingkup Web Development," vol. 01, no. 6, 2022.
- [42] A. Mubariz, D. Nur, E. Tungadi, and M. N. Y. Utomo, "Perancangan Back-End Server Menggunakan Arsitektur Rest dan Platform Node.JS (Studi Kasus: Sistem Pendaftaran Ujian Masuk Politeknik Negeri Ujung Pandang)," 2020.
- [43] S. N. Yanti and E. Rihyanti, "Penerapan Rest API untuk Sistem Informasi Film Secara Daring," *JIUP*, vol. 6, no. 1, p. 195, Mar. 2021, doi: [10.32493/informatika.v6i1.10033](https://doi.org/10.32493/informatika.v6i1.10033).
- [44] L. A. Abdillah, "PERANCANGAN BASISDATA SISTEM INFORMASI PENGAJIAN".
- [45] S. Tyowati and R. Irawan, "Implementasi Framework Codeigniter Untuk

- Pengembangan Website Pada Dinas Perkebunan Provinsi Kalimantan Tengah," *saintekom*, vol. 7, no. 1, p. 67, Apr. 2017, doi: [10.33020/saintekom.v7i1.22](https://doi.org/10.33020/saintekom.v7i1.22).
- [46] F. F. Nursaid, A. H. Brata, and A. P. Kharisma, "Pengembangan Sistem Informasi Pengelolaan Persediaan Barang Dengan ReactJS Dan React Native Menggunakan Prototype (Studi Kasus : Toko Uda Fajri)".
- [47] "Kanban - A brief introduction | Atlassian." <https://www.atlassian.com/agile/kanban> (accessed Apr. 13, 2023).
- [48] K. Graham, "TechMatters: Getting on the 'Kanban'-wagon: Using KanbanFlow for Time and Project Management," vol. 43, no. 3.
- [49] M. O. Ahmad, D. Dennehy, K. Conboy, and M. Oivo, "Kanban in software engineering: A systematic mapping study," *Journal of Systems and Software*, vol. 137, pp. 96–113, 2018, doi: <https://doi.org/10.1016/j.jss.2017.11.045>.
- [50] Alaidaros, Hamzah & Omar, Mazni & Romli, Rohaida. (2021). The State of the Art of Agile Kanban Method: Challenges and Opportunities. *Independent Journal of Management & Production*. 12. 10.14807/ijmp.v12i8.1482.
- [51] Handayani Akar, Rutmeida. (2021). Literature Review: Kelebihan Pengujian Kotak Hitam (Black Box Testing) Pada Pengujian Perangkat Lunak.
- [52] "IoT Connectivity: Choosing a Black-Box, White-Box, or Gray-Box Approach - Embedded Computing Design." <https://embeddedcomputing.com/technology/IoT/IoT-connectivity-choosing-a-black-box-white-box-or-gray-box-approach> (accessed Apr. 14, 2023).
- [53] Brooke, John. (2013). SUS: a retrospective. *Journal of Usability Studies*. 8. 29-40.
- [54] F. G. Sembodo, G. F. Fitriana, and N. A. Prasetyo, "Evaluasi Usability Website Shopee Menggunakan System Usability Scale (SUS)," *JAIC*, vol. 5, no. 2, pp. 146–150, Nov. 2021, doi: [10.30871/jaic.v5i2.3293](https://doi.org/10.30871/jaic.v5i2.3293).
- [55] D. P. Kesuma, "Penggunaan Metode System Usability Scale Untuk Mengukur Aspek Usability Pada Media Pembelajaran Daring di Universitas XYZ," *JATISI*, vol. 8, no. 3, pp. 1615–1626, Sep. 2021, doi: [10.35957/jatisi.v8i3.1356](https://doi.org/10.35957/jatisi.v8i3.1356).

LAMPIRAN

Lampiran 1 Kode Program

Link Github :

- Frontend : <https://github.com/dhifafaz/website-iterahero>
- Backend : https://github.com/Toolop/iterahero_backend
- Figma : <https://www.figma.com/file/9MnOASY4doQ11GUbJ1F7r/Ini-website-monitoring-controlling?node-id=70%3A3776&t=wHAXERw6zOjOmDW8-0>
- Sistem tertanam : https://github.com/pramudya0406/CHC_CoolingPad.git

Lampiran 2 Board Agile Kanban

Epic Backend

| No | Story | Status |
|----|--|--------|
| 1 | Init Backend | done |
| 2 | BE-Endpoint Get Data sensor | done |
| 3 | BE-Endpoint Edit Actuator | done |
| 4 | BE-Endpoint get Actuator Detail | done |
| 5 | BE-Endpoint get Actuator by <i>greenhouse</i> | done |
| 6 | BE-Endpoint upload actuator | done |
| 7 | BE-Endpoint upload Notifikasi | done |
| 8 | BE-Endpoint update Sensor | done |
| 9 | BE-Membuat endpoint get <i>Greenhouse</i> | done |
| 10 | BE-Membuat endpoint Menambah <i>Greenhouse</i> | done |
| 11 | BE-Membuat endpoint Register | done |
| 12 | BE-Membuat endpoint login | done |
| 13 | BE-Membuat endpoint get <i>Greenhouse</i> Detail | done |
| 14 | BE-Membuat endpoint update <i>Greenhouse</i> | done |
| 15 | BE-Membuat endpoint Hapus <i>Greenhouse</i> | done |
| 16 | BE-Endpoint tambah Sensor | done |
| 17 | BE-Endpoint get Sensor by <i>greenhouse</i> | done |
| 18 | BE-Endpoint get detail sensor | done |
| 19 | BE-Endpoint Get notifikasi by user | done |
| 20 | BE-Endpoint Hapus Notifikasi | done |
| 21 | BE-Endpoint Delete Actuator | done |
| 22 | BE-Endpoint upload Data Sensor dari alat | done |
| 23 | Endpoint status Sensor offline online | done |
| 24 | BE-Endpoint kontrol actuator | done |
| 25 | BE-Endpoint status Actuator offline online | done |
| 26 | BE-Endpoint Automation by Sensor | done |
| 27 | BE-Endpoint Automation by Time | done |
| 28 | BE-Endpoint Riwayat Actuator | done |
| 29 | BE-Endpoint summary | done |
| 30 | BE-Endpoint automation for server | done |

Epic Frontend

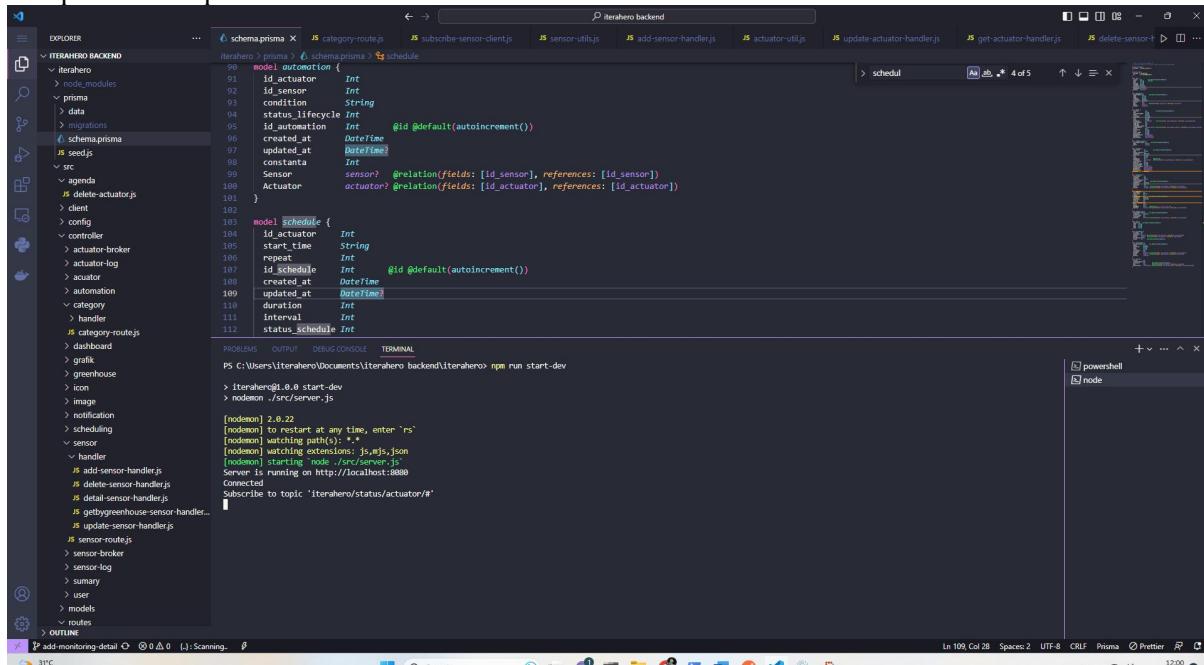
| No | Story | Status |
|----|--|--------|
| 1 | FE-Membuat Halaman Login | done |
| 2 | FE-Halaman Dashboard | done |
| 3 | FE-Halaman Menampilkan <i>greenhouse</i> | done |
| 4 | FE-Halaman Menambah <i>greenhouse</i> | done |
| 5 | FE-Halaman mengedit <i>Greenhouse</i> | done |
| 6 | FE-Halaman hapus <i>greenhouse</i> | done |
| 7 | FE-Halaman menampilkan Sensor | done |
| 8 | FE-Halaman menambah sensor | done |
| 9 | FE-Halaman update sensor | done |

| | | |
|----|-----------------------------------|------|
| 10 | FE-Halaman delete Sensor | done |
| 11 | FE-Halaman menampilkan aktuator | done |
| 12 | FE-Halaman tambah aktuator | done |
| 13 | FE-Halaman edit Actuator | done |
| 14 | FE-Halaman hapus Actuator | done |
| 15 | FE-Halaman Notifikasi | done |
| 16 | FE-Halaman Hapus Notifikasi | done |
| 17 | FE-Halaman data sensor ke website | done |
| 18 | FE-Halaman kendali actuator | done |
| 19 | FE- Halaman Grafik Sensor | done |
| 20 | FE-Halaman Otomatis | done |
| 21 | FE-Halaman History Actuator | done |
| 22 | FE-Halaman Otomatis By Sensor | done |
| 23 | FE-Halaman Otomatis by TIime | done |
| 24 | FE-Fix to hosting | done |

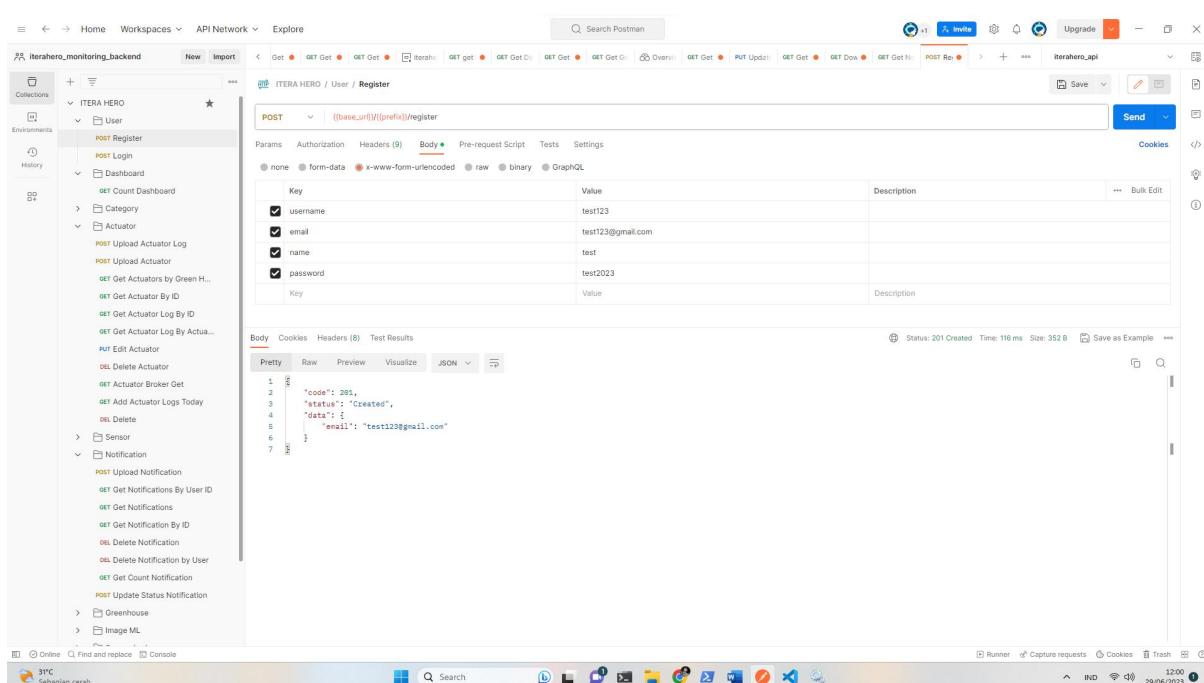
Epic Sistem tertanam

| No | Story | Status |
|----|--------------------------|--------|
| 1 | T-Monitoring Lingkungan | done |
| 2 | T-Kontrol Exhaust Fan | done |
| 3 | T-Kontrol Cooling Pad | done |
| 4 | T-Sensor Debit | done |
| 5 | T-Kontrol Selenoid Valve | done |
| 6 | T-Kontrol Pompa | done |

Lampiran 3 Lampiran Pembuatan Backend



The screenshot shows a code editor with the file `schema.prisma` open. The code defines two models: `automation` and `schedule`. The `automation` model has fields for `id_actuator`, `id_sensor`, `condition`, `status.lifecycle`, `id_automation`, `created_at`, `updated_at`, and `constant`. The `schedule` model has fields for `id_actuator`, `start_time`, `end_time`, `id_schedule`, `created_at`, `updated_at`, `duration`, `interval`, and `status_schedule`. A terminal window below shows the command `npm run start-dev` being run, and the output indicates the server is running on `http://localhost:8080`.



The screenshot shows the Postman application interface. A POST request is being prepared to the endpoint `/User/Register`. The request body contains the following JSON:

```

{
  "username": "test123",
  "email": "test123@gmail.com",
  "name": "test",
  "password": "test2023"
}

```

The response status is 201 Created, with a response time of 116 ms and a size of 352 B.

ITERA HERO / User / Login

POST `((base_url))/((prefix))/Login`

Params Authorization Headers (9) Body Pre-request Script Tests Settings

Body

| Key | Value | Description |
|--|-------------------------|-------------|
| <input checked="" type="checkbox"/> email | iterahero2022@gmail.com | |
| <input checked="" type="checkbox"/> password | iterahero2022 | |

Body Cookies Headers (8) Test Results

```

1
2 "code": 200,
3 "status": "Ok",
4 "data": [
5   {
6     "email": "iterahero2022@gmail.com",
7     "accessToken": "eyJhbGciOiJIUzI1NiBhInRScC16IkpxJ3AZ21hakuvY29tIiawRfdXlciMsviaWF0IjoxNjg4MDE0ODQgIiQ7iGRMhUwcdVvOHkSe74uP0gIGNs"
8   }
9 ]

```

Status: 200 OK Time: 112 ms Size: 533 B Save as Example

ITERA HERO / Greenhouse / Get Green House By User ID

GET `((base_url))/((prefix))/greenhouse`

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

| Key | Value | Description |
|-----|-------|-------------|
| Key | Value | Description |

Body Cookies Headers (13) Test Results

```

1
2 "code": 200,
3 "status": "Ok",
4 "data": [
5   {
6     "id": 1,
7     "name": "Greenhouse ITERA",
8     "image": "https://res.cloudinary.com/apolaporan/image/upload/v1668391844/greenhouse_images/nletgsenw10fj7xztzl.jpg",
9     "location": "Tambang Selatan",
10     "create_at": "2023-06-28T10:42:10.833Z",
11     "user_id": 1
12   }
13 ]

```

Status: 200 OK Time: 1670 ms Size: 700 B Save as Example

Lampiran 4 Pemasangan alat pada *smart greenhouse*





Lampiran 5 Pengujian alat



Lampiran 6 Pengujian SUS

SUS Sistem Monitoring Greenhouse berbasis IoT

data-data yang diambil untuk kepentingan dari penelitian tugas akhir saya yang berjudul sistem monitoring greenhouse, mohon bantuananya untuk mengisi dengan sungguh-sungguh

The respondent's email (lp3m@itera.ac.id) was recorded on submission of this form.

* Indicates required question

Nama *

Zunanik Mufidah

Pekerjaan *

Install react.js - Penelusuran Google
google.com/search?q=install+react+js&...
Introduction | Learn Next.js
nextjs.org/learn/foundations/about-nextjs
Getting Started - React
legacy.reactjs.org/docs/getting-started...
From JavaScript to React | Learn Nex...
nextjs.org/learn/foundations/from-javas...

Lampiran 7 melon metode konvensional



Lampiran 8 melon metode Smart Farming

