

**BUKU TUGAS AKHIR  
CAPSTONE DESIGN**



**Pengembangan Sistem Monitoring Tel-U Interactive Food  
Assistant (TIFA) dalam Upaya Peningkatan Pelayanan  
di Tel-U Coffee**

**Oleh:**  
**Anda Figo Haq/1103213098**  
**Muhammad Jibran Hady/1103210132**  
**Nabila Ardiyani/1103213029**

**PROGRAM STUDI S1 TEKNIK KOMPUTER  
FAKULTAS TEKNIK ELEKTRO  
UNIVERSITAS TELKOM  
BANDUNG  
2025**

# **LEMBAR PENGESAHAN BUKU CAPSTONE DESIGN**

**PENGEMBANGAN SISTEM MONITORING TEL-U INTERACTIVE FOOD  
ASSISTANT (TIFA) DALAM UPAYA PENINGKATAN PELAYANAN  
DI TEL-U COFFEE**

*DEVELOPMENT OF THE MONITORING SYSTEM FOR TEL-U INTERACTIVE FOOD  
ASSISTANT (TIFA) TO IMPROVE SERVICE QUALITY AT TEL-U COFFEE*

**Telah disetujui dan disahkan sebagai bagian dari Capstone Design  
Program Studi S1 Teknik Komputer  
Fakultas Teknik Elektro  
Universitas Telkom  
Bandung**

**Disusun oleh:**

**Anda Figo Haq/1103213098**

**Muhammad Jibran Hady/1103210132**

**Nabila Ardiyani/1103213029**

**Bandung, 7 Juli 2025**

**Menyetujui,**

Pembimbing 1



Faisal Candrasyah Hasibuan, S.T., M.T.

NIP. 20910005

Pembimbing 2



R Rogers Dwiputra Setiady, S.T., M.T.

NIP. 219185996

## **LEMBAR PERNYATAAN ORISINALITAS**

Saya, yang bertanda tangan di bawah ini

Nama : Anda Figo Haq  
NIM : 1103213098  
Alamat : The GREEN HILL C5 No. 1 Cibinong, Kab.Bogor, Jawa Barat  
No. Telepon : 082133360116  
Email : andahaq@gmail.com

Menyatakan bahwa Buku Capstone Design ini merupakan karya orisinal saya sendiri bersama dengan kelompok Capstone Design saya, dengan judul:

### **PENGEMBANGAN SISTEM MONITORING TEL-U INTERACTIVE FOOD ASSISTANT (TIFA) DALAM UPAYA PENINGKATAN PELAYANAN DI TEL-U COFFEE**

*DEVELOPMENT OF THE MONITORING SYSTEM FOR TEL-U INTERACTIVE FOOD  
ASSISTANT (TIFA) TO IMPROVE SERVICE QUALITY AT TEL-U COFFEE*

Atas pernyataan ini, saya siap menanggung resiko/sanksi yang dijatuhan kepada saya apabila dikemudian ditemukan adanya pelanggaran terhadap kejujuran akademik atau etika keilmuan dalam karya ini, atau ditemukan bukti yang menunjukkan ketidak aslian karya ini.

Bandung, 23 Juni 2025



---

Anda Figo Haq  
1103213098

## **LEMBAR PERNYATAAN ORISINALITAS**

Saya, yang bertanda tangan di bawah ini

Nama : Muhammad Jibran Hady  
NIM : 1103210132  
Alamat : Jalan Melati Raya No. 333 BTN Rembiga, Kota Mataram, NTB  
No. Telepon : 0895336176053  
Email : m.jibranhady@gmail.com

Menyatakan bahwa Buku Capstone Design ini merupakan karya orisinal saya sendiri bersama dengan kelompok Capstone Design saya, dengan judul:

**PENGEMBANGAN SISTEM MONITORING TEL-U INTERACTIVE FOOD  
ASSISTANT (TIFA) DALAM UPAYA PENINGKATAN PELAYANAN  
DI TEL-U COFFEE**

*DEVELOPMENT OF THE MONITORING SYSTEM FOR TEL-U INTERACTIVE FOOD  
ASSISTANT (TIFA) TO IMPROVE SERVICE QUALITY AT TEL-U COFFEE*

Atas pernyataan ini, saya siap menanggung resiko/sanksi yang dijatuhkan kepada saya apabila dikemudian ditemukan adanya pelanggaran terhadap kejujuran akademik atau etika keilmuan dalam karya ini, atau ditemukan bukti yang menunjukkan ketidak aslian karya ini.

Bandung, 23 Juni 2025



Muhammad Jibran Hady  
1103210132

## **LEMBAR PERNYATAAN ORISINALITAS**

Saya, yang bertanda tangan di bawah ini

Nama : Nabila Ardiyani  
NIM : 1103213029  
Alamat : Dk. Aglik 1, RT 02/RW 04, Lembupurwo, Mirit, Kebumen  
No. Telepon : 082136109519  
Email : nabilaardiyani14@gmail.com

Menyatakan bahwa Buku Capstone Design ini merupakan karya orisinal saya sendiri bersama dengan kelompok Capstone Design saya, dengan judul:

### **PENGEMBANGAN SISTEM MONITORING TEL-U INTERACTIVE FOOD ASSISTANT (TIFA) DALAM UPAYA PENINGKATAN PELAYANAN DI TEL-U COFFEE**

*DEVELOPMENT OF THE MONITORING SYSTEM FOR TEL-U INTERACTIVE FOOD  
ASSISTANT (TIFA) TO IMPROVE SERVICE QUALITY AT TEL-U COFFEE*

Atas pernyataan ini, saya siap menanggung resiko/sanksi yang dijatuhkan kepada saya apabila dikemudian ditemukan adanya pelanggaran terhadap kejujuran akademik atau etika keilmuan dalam karya ini, atau ditemukan bukti yang menunjukkan ketidak aslian karya ini.

Bandung, 23 Juni 2025



---

Nabila Ardiyani  
1103213029

## ABSTRAK

Tel-U Coffee menghadapi tantangan operasional yang signifikan selama jam sibuk, termasuk peningkatan pesat jumlah pesanan, keterbatasan staf, keterlambatan pengiriman, kesalahan pesanan, dan penurunan kepuasan pelanggan sebesar 15%. Upaya sebelumnya menggunakan robot pengantar makanan generasi awal terbukti tidak memadai karena keterbatasan teknis, seperti antarmuka pengguna sederhana berbasis App Inventor dan tidak adanya sistem monitoring *real-time*, sehingga tidak mampu mengelola lonjakan permintaan secara optimal. Kondisi ini memerlukan solusi inovatif yang mengatasi kelemahan teknis sistem sebelumnya sambil memenuhi kebutuhan operasional kafe dalam menjaga kualitas pelayanan dan efisiensi. Penelitian ini mengembangkan sistem monitoring berbasis *Internet of Things* (IoT) yang terintegrasi dengan robot pengantar makanan. Sistem ini terdiri dari aplikasi Android dan web monitoring yang memungkinkan barista memberikan perintah dan memantau status robot secara *real-time*, meliputi lokasi, kapasitas baterai, dan status *tray*. Teknologi yang digunakan mencakup ESP32 sebagai mikrokontroler, protokol MQTT untuk komunikasi data, dan MySQL sebagai basis data. Sistem dirancang agar mudah dioperasikan tanpa mengganggu alur kerja barista dan memberikan transparansi dalam proses layanan. Metodologi pengembangan meliputi analisis kebutuhan, perancangan sistem, integrasi perangkat lunak dan keras, serta pengujian menggunakan *User Acceptance Testing* (UAT). Implementasi Tel-U Interactive Food Assistant (TIFA) berhasil mengatasi tantangan operasional Tel-U Coffee selama jam sibuk. Sistem ini terbukti efektif meningkatkan efisiensi pelayanan melalui otomatisasi pengantaran dan pemantauan pesanan, dengan tingkat kepuasan pengguna mencapai 90,73% untuk aplikasi Android dan 92% untuk web monitoring, serta mampu menangani lonjakan pesanan hingga 40%. Keterbatasan sistem sebelumnya telah diatasi melalui aplikasi Android yang lebih responsif dan integrasi web monitoring terpusat. Meskipun demikian, sistem ini memiliki keterbatasan dalam hal jumlah responden pengujian dan belum diujinya skenario *multi-robot* atau integrasi dengan sistem POS. Pengembangan selanjutnya akan fokus pada peningkatan skalabilitas, keandalan, dan penambahan fitur AI interaktif serta dukungan *multi-robot*. TIFA menunjukkan potensi sebagai solusi inovatif untuk meningkatkan pelayanan industri kafe dan dapat menjadi model penerapan teknologi pintar di sektor serupa.

Kata Kunci: *Internet of Things* (IoT), robot pengantar makanan, sistem monitoring *real-time*

## ***ABSTRACT***

*Tel-U Coffee faces significant operational challenges during peak hours, including rapid increases in order volumes, staff limitations, delivery delays, order errors, and a 15% decline in customer satisfaction. Previous efforts using first-generation food delivery robots proved inadequate due to technical limitations, such as simple user interfaces based on App Inventor and the absence of real-time monitoring systems, resulting in suboptimal handling of demand surges. This situation necessitates innovative solutions that address the technical weaknesses of previous systems while meeting the café's operational needs to maintain service quality and efficiency. This research develops an Internet of Things (IoT)-based monitoring system integrated with food delivery robots. The system consists of an Android application and web monitoring platform that enables baristas to issue commands and monitor robot status in real-time, including location, battery capacity, and tray status. The technologies employed include ESP32 as the microcontroller, MQTT protocol for data communication, and MySQL as the database. The system is designed to be easily operated without disrupting barista workflows while providing transparency in the service process. The development methodology encompasses requirements analysis, system Design, software and hardware integration, and Testing using User Acceptance Testing (UAT). The implementation of Tel-U Interactive Food Assistant (TIFA) successfully addresses Tel-U Coffee's operational challenges during peak hours. The system proves effective in improving service efficiency through delivery automation and order monitoring, achieving user satisfaction rates of 90.73% for the Android application and 92% for web monitoring, while capable of handling order surges up to 40%. Previous system limitations have been overcome through a more responsive Android application and centralized web monitoring integration. Nevertheless, the system has limitations regarding the number of testing respondents and untested multi-robot scenarios or POS system integration. Future development will focus on enhancing scalability, reliability, and adding interactive AI features along with multi-robot support. TIFA demonstrates significant potential as an innovative solution for improving café industry services and can serve as a model for smart technology implementation in similar sectors.*

*Keywords:* Food delivery robot, Internet of Things (IoT), real-time monitoring system

## KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Allah SWT atas segala rahmat, hidayah, dan karunia-Nya, sehingga kami dapat menyelesaikan penulisan tugas akhir dengan judul "PENGEMBANGAN SISTEM MONITORING TEL-U INTERACTIVE FOOD ASSISTANT (TIFA) DALAM UPAYA PENINGKATAN PELAYANAN DI TEL-U COFFEE". Tugas akhir ini merupakan bagian akhir dari perjalanan studi kami di Program Studi S1 Teknik Komputer, Fakultas Teknik Elektro, Telkom University.

Kami ingin menyampaikan rasa hormat dan terima kasih yang sebesar-besarnya kepada dosen pembimbing kami yang telah memberikan bimbingan, arahan, dan dukungan penuh selama proses penyusunan tugas akhir ini. Ucapan terima kasih juga kami sampaikan kepada seluruh dosen yang telah berbagi ilmu dan pengalaman berharga selama masa studi kami. Tak lupa, kami mengucapkan terima kasih yang tulus kepada keluarga kami yang senantiasa memberikan doa, semangat, dan motivasi dalam setiap langkah kami. Tanpa dukungan mereka, kami tidak akan mampu mencapai titik ini.

Dalam proses penyusunan tugas akhir ini, kami menghadapi berbagai tantangan dan hambatan. Namun, pengalaman tersebut telah memberikan pelajaran yang sangat berharga yang dapat kami jadikan bekal di dunia kerja maupun kehidupan sehari-hari. Melalui tugas akhir ini, kami berharap dapat memberikan kontribusi positif terhadap pengembangan teknologi dan pelayanan di lingkungan kampus, khususnya di Tel-U Coffee. Kami menyadari bahwa tugas akhir ini masih jauh dari sempurna. Oleh karena itu, kami sangat mengharapkan kritik dan saran yang membangun dari para pembaca demi penyempurnaan di masa yang akan datang. Semoga karya ini dapat menjadi referensi yang bermanfaat bagi penelitian selanjutnya dan memberikan dampak positif dalam pengembangan sistem monitoring berbasis *Internet of Things* (IoT).

Akhir kata, semoga tugas akhir ini memberikan manfaat dan menjadi amal jariyah bagi semua pihak yang terlibat. Semoga Allah SWT senantiasa melimpahkan rahmat dan hidayah-Nya kepada kita semua. Aamiin.

## UCAPAN TERIMAKASIH

Segala puji dan syukur penulis panjatkan ke hadirat Allah SWT atas limpahan rahmat, hidayah, dan karunia-Nya yang tiada henti. Atas izin dan ridho-Nya, penulis dapat menyelesaikan Tugas Akhir yang berjudul **“PENGEMBANGAN SISTEM MONITORING TEL-U INTERACTIVE FOOD ASSISTANT (TIFA) DALAM UPAYA PENINGKATAN PELAYANAN DI TEL-U COFFEE”** ini dengan baik. Tanpa bimbingan dan kekuatan dari-Nya, penyusunan Tugas Akhir ini tidak akan mungkin tercapai. Pada kesempatan ini, penulis ingin mengucapkan terima kasih yang mendalam kepada berbagai pihak yang telah memberikan dukungan dan bimbingan selama proses penyusunan Tugas Akhir ini:

1. Kepada Allah SWT, yang selalu memberikan kekuatan, kesabaran, dan kemudahan dalam setiap langkah penulis. Segala kemuliaan dan puji hanya milik-Nya.
2. Kedua orang tua penulis pertama, kedua, dan ketiga yang senantiasa mendoakan, memberikan dukungan moral, dan kasih sayang tanpa batas. Tanpa doa dan dorongan mereka, penulis tidak akan berada di titik ini.
3. Bapak Faisal Chandrasyah Hasibuan, S.T., M.T. selaku dosen pembimbing 1, yang dengan sabar memberikan bimbingan, masukan, serta arahan yang sangat berarti dalam penyusunan Tugas Akhir ini.
4. Bapak R Rogers Setiady, S.T., M.T. selaku dosen pembimbing 2, yang dengan sabar memberikan bimbingan, masukan, serta arahan yang sangat berarti dalam penyusunan Tugas Akhir ini.
5. Bapak Randy Erfa Saputra, ST., M.T., dan Bapak Roswan Latuconsina S.T., M.T., selaku dosen wali kami yang dengan sabar memberikan masukan serta arahan yang sangat berarti.
6. Seluruh dosen di Program Studi S1 Teknik Komputer, Universitas Telkom, yang telah memberikan ilmu dan pengetahuan yang menjadi dasar dalam menyelesaikan tugas akhir ini.
7. Rifqy Fachrizi, S.T. selaku *Project Manager* Tim Robot TIFA, serta seluruh anggota Tim Robot TIFA, atas waktu, tenaga, dan pemikiran yang telah dicurahkan dalam

proses pengembangan dan pembentukan Robot TIFA. Kontribusi dan kerja sama yang diberikan sangat berarti dalam mendukung keberhasilan penelitian ini.

Semoga Tugas Akhir ini dapat memberikan manfaat bagi semua pihak yang membacanya dan menjadi sumbangsih yang berharga bagi perkembangan ilmu pengetahuan.

## DAFTAR ISI

<b>LEMBAR PENGESAHAN .....</b>	<b>ii</b>
<b>BUKU CAPSTONE DESIGN.....</b>	<b>ii</b>
<b>LEMBAR PERNYATAAN ORISINALITAS.....</b>	<b>iii</b>
<b>LEMBAR PERNYATAAN ORISINALITAS.....</b>	<b>iv</b>
<b>LEMBAR PERNYATAAN ORISINALITAS.....</b>	<b>v</b>
<b>ABSTRAK .....</b>	<b>vi</b>
<b>ABSTRACT.....</b>	<b>vii</b>
<b>KATA PENGANTAR.....</b>	<b>viii</b>
<b>UCAPAN TERIMAKASIH .....</b>	<b>ix</b>
<b>DAFTAR ISI.....</b>	<b>xi</b>
<b>DAFTAR GAMBAR.....</b>	<b>xv</b>
<b>DAFTAR TABEL .....</b>	<b>xviii</b>
<b>BAB 1 USULAN GAGASAN.....</b>	<b>1</b>
1.1 <b>Deskripsi Umum Masalah dan Kebutuhan .....</b>	<b>1</b>
1.2 <b>Analisis Masalah .....</b>	<b>4</b>
1.2.1 <b>Aspek Teknis .....</b>	<b>4</b>
1.2.2 <b>Aspek Sumber Daya Manusia.....</b>	<b>5</b>
1.2.3 <b>Aspek Ekonomi .....</b>	<b>5</b>
1.3 <b>Analisis Solusi yang Ada .....</b>	<b>5</b>
1.4 <b>Kesimpulan dan Ringkasan CD-1 .....</b>	<b>7</b>
<b>BAB 2 TINJAUAN PUSTAKA .....</b>	<b>9</b>
2.1 <b>Dasar Penentuan Spesifikasi.....</b>	<b>9</b>
2.2 <b>Daftar Batasan dan Spesifikasi.....</b>	<b>11</b>
2.2.1 <b>Daftar Batasan dan Spesifikasi Robot .....</b>	<b>11</b>
2.2.2 <b>Daftar Batasan dan Spesifikasi Aplikasi dan Web Monitoring .....</b>	<b>11</b>
2.2.3 <b>Batasan Sistem .....</b>	<b>12</b>

<b>2.3</b>	<b>Pengukuran/Verifikasi Spesifikasi .....</b>	<b>13</b>
<b>2.4</b>	<b>Kesimpulan dan Ringkasan CD-2 .....</b>	<b>14</b>
<b>BAB 3 SPESIFIKASI DAN DESAIN SISTEM .....</b>		<b>16</b>
<b>3.1</b>	<b>Alternatif Usulan Solusi .....</b>	<b>16</b>
<b>3.1.1</b>	<b>Usulan Solusi Mobile Development Framework.....</b>	<b>16</b>
<b>3.1.2</b>	<b>Usulan Solusi Database.....</b>	<b>17</b>
<b>3.1.3</b>	<b>Usulan Solusi Mikrokontroller .....</b>	<b>19</b>
<b>3.1.4</b>	<b>Usulan Solusi Backend Framework .....</b>	<b>20</b>
<b>3.1.5</b>	<b>Usulan Solusi Sensor Pendekripsi Makanan.....</b>	<b>22</b>
<b>3.1.6</b>	<b>Usulan Solusi Device .....</b>	<b>23</b>
<b>3.1.7</b>	<b>Usulan Solusi Framework Frontend Web .....</b>	<b>24</b>
<b>3.1.8</b>	<b>Usulan Solusi Monitoring Tegangan.....</b>	<b>26</b>
<b>3.2</b>	<b>Analisis dan Pemilihan Solusi .....</b>	<b>27</b>
<b>3.2.1</b>	<b>Parameter Pemilihan Solusi.....</b>	<b>28</b>
<b>3.3</b>	<b>Desain Solusi Terpilih.....</b>	<b>38</b>
<b>3.3.1</b>	<b>Arsitektur Rancangan Umum .....</b>	<b>38</b>
<b>3.3.2</b>	<b>Arsitektur Modul IoT TIFA .....</b>	<b>40</b>
<b>3.3.3</b>	<b>Desain Flow Diagram Fungsional.....</b>	<b>42</b>
<b>3.3.4</b>	<b>Entity Relationship Diagram Database .....</b>	<b>43</b>
<b>3.3.5</b>	<b>Flowchart .....</b>	<b>45</b>
<b>3.3.6</b>	<b>UML (Unified Modeling Language).....</b>	<b>49</b>
<b>3.3.7</b>	<b>Activity Diagram.....</b>	<b>52</b>
<b>3.3.8</b>	<b>Sequence.....</b>	<b>60</b>
<b>3.3.9</b>	<b>Desain UI/UX.....</b>	<b>67</b>
<b>3.4</b>	<b>Jadwal dan Anggaran.....</b>	<b>78</b>
<b>3.4.1</b>	<b>Jadwal .....</b>	<b>78</b>
<b>3.4.2</b>	<b>Anggaran .....</b>	<b>81</b>

<b>BAB 4 IMPLEMENTASI .....</b>	<b>83</b>
<b>4.1    Deskripsi Umum Implementasi .....</b>	<b>83</b>
<b>4.1.1    Pengembangan Aplikasi TIFA .....</b>	<b>84</b>
<b>4.1.2    Pengembangan Web Monitoring.....</b>	<b>84</b>
<b>4.2    Detail Implementasi.....</b>	<b>85</b>
<b>4.2.1    Detil Implemetasi Aplikasi TIFA .....</b>	<b>85</b>
<b>4.2.2    Sub-Sistem Backend API .....</b>	<b>87</b>
<b>4.2.3    Sub-Sistem AntarRepository.kt.....</b>	<b>95</b>
<b>4.2.4    Sub-Sistem AntarViewModel.kt.....</b>	<b>98</b>
<b>4.2.5    Sub-Sistem Struktur Data: AntarData.kt dan OrderData.kt.....</b>	<b>101</b>
<b>4.2.6    Sub-Sistem Tampilan UI Mode Antar .....</b>	<b>102</b>
<b>4.2.7    Hasil Implementasi .....</b>	<b>113</b>
<b>4.2.8    Penggunaan Hardware untuk Aplikasi .....</b>	<b>113</b>
<b>4.2.9    Detil Implementasi Web Monitoring .....</b>	<b>114</b>
<b>4.2.10   Data Layer (Database).....</b>	<b>118</b>
<b>4.2.11   Application Layer (Backend).....</b>	<b>121</b>
<b>4.2.12   Presentation Layer (Frontend) .....</b>	<b>131</b>
<b>4.2.13   Hasil Implementasi .....</b>	<b>143</b>
<b>4.2.14   Penggunaan Hardware Untuk Web .....</b>	<b>143</b>
<b>Tabel 4. 2 Penggunaan Hardware Web Monitoring .....</b>	<b>143</b>
<b>4.2.15   Detil Implementasi Monitoring Sensor IR ke UI .....</b>	<b>144</b>
<b>4.2.16   Layer Pemrosesan Status Konfirmasi pada Backend PHP .....</b>	<b>144</b>
<b>4.3    Prosedur Pengoperasian.....</b>	<b>146</b>
<b>4.3.1    Prosedur Pengoperasian untuk Aplikasi TIFA.....</b>	<b>146</b>
<b>4.3.2    Prosedur Pengoperasian untuk Web Monitoring.....</b>	<b>152</b>
<b>4.3.3    Penyerahan Produk kepada Mitra.....</b>	<b>161</b>
<b>BAB 5 PENGUJIAN .....</b>	<b>162</b>

<b>5.1</b>	<b>Skenario Umum Pengujian .....</b>	<b>162</b>
<b>5.1.1</b>	<b>Security Testing.....</b>	<b>162</b>
<b>5.1.2</b>	<b>Integration Testing.....</b>	<b>163</b>
<b>5.1.3</b>	<b>Alpha Testing .....</b>	<b>163</b>
<b>5.1.4</b>	<b>User Acceptance Testing (UAT) .....</b>	<b>163</b>
<b>5.2</b>	<b>Detil Pengujian .....</b>	<b>163</b>
<b>5.2.1</b>	<b>Security Testing.....</b>	<b>164</b>
<b>5.2.2</b>	<b>Integration Testing.....</b>	<b>165</b>
<b>5.2.3</b>	<b>Alpha Testing .....</b>	<b>181</b>
<b>5.2.4</b>	<b>User Acceptance Testing (UAT) .....</b>	<b>201</b>
<b>5.3</b>	<b>Analisa Hasil Pengujian .....</b>	<b>208</b>
<b>5.3.1</b>	<b>Security Testing.....</b>	<b>208</b>
<b>5.3.2</b>	<b>Integration Testing.....</b>	<b>210</b>
<b>5.3.3</b>	<b>Alpha Testing .....</b>	<b>212</b>
<b>5.3.4</b>	<b>User Acceptance Testing (UAT) .....</b>	<b>214</b>
<b>5.4</b>	<b>Kesimpulan.....</b>	<b>219</b>
<b>DAFTAR PUSTAKA.....</b>		<b>220</b>
<b>LAMPIRAN.....</b>		<b>227</b>

## DAFTAR GAMBAR

<b>Gambar 1. 1 Grafik Perbandingan Jumlah Pesanan pada <i>Peak hours</i> dan <i>Non-Peak hours</i> di Tel-U Coffee.....</b>	<b>1</b>
<b>Gambar 1. 2 Grafik Perbandingan Tingkat Kepuasan Pelanggan pada <i>Peak hours</i> dan <i>Non-Peak hours</i> di Tel-U Coffee .....</b>	<b>2</b>
<b>Gambar 3. 1 ESP32S3.....</b>	<b>33</b>
<b>Gambar 3. 2 Tablet.....</b>	<b>35</b>
<b>Gambar 3. 3 IR Sensor .....</b>	<b>36</b>
<b>Gambar 3. 4 <i>Voltage sensor</i>.....</b>	<b>38</b>
<b>Gambar 3. 5 Arsitektur Rancangan Umum .....</b>	<b>38</b>
<b>Gambar 3. 6 Arsitektur Rancangan Modul IoT .....</b>	<b>40</b>
<b>Gambar 3. 7 Desain <i>Flow Diagram</i> Fungsional .....</b>	<b>42</b>
<b>Gambar 3. 8 Entity Relationship Diagram Database .....</b>	<b>43</b>
<b>Gambar 3. 9 <i>Flowchart</i> Aplikasi.....</b>	<b>47</b>
<b>Gambar 3. 10 <i>Flowchart</i> Web Monitoring .....</b>	<b>49</b>
<b>Gambar 3. 11 Use Case Aplikasi .....</b>	<b>50</b>
<b>Gambar 3. 12 Use Case Diagram Web Monitoring .....</b>	<b>52</b>
<b>Gambar 3. 13 Activity Diagram Aplikasi fitur Media.....</b>	<b>53</b>
<b>Gambar 3. 14 Activity Diagram Aplikasi Mode Antar .....</b>	<b>53</b>
<b>Gambar 3. 15 Activity Diagram Aplikasi Fitur Peta.....</b>	<b>54</b>
<b>Gambar 3. 16 Activity Diagram Aplikasi Fitur <i>TalkBack</i> .....</b>	<b>55</b>
<b>Gambar 3. 17 Activity Diagram Web Monitoring Fitur <i>Login</i> .....</b>	<b>55</b>
<b>Gambar 3. 18 Activity Diagram Web Monitoring Fitur Daftar Robot .....</b>	<b>57</b>
<b>Gambar 3. 19 Activity Diagram Web Monitoring Fitur Tambah Robot .....</b>	<b>58</b>
<b>Gambar 3. 20 Activity Diagram Web Monitoring Fitur Tambah Jenis Robot .....</b>	<b>58</b>
<b>Gambar 3. 21 Activity Diagram Web Monitoring Fitur Daftar Jenis Robot.....</b>	<b>60</b>
<b>Gambar 3. 22 Sequence Aplikasi Media .....</b>	<b>61</b>
<b>Gambar 3. 23 Sequence Aplikasi Mode Antar .....</b>	<b>61</b>
<b>Gambar 3. 24 Sequence Aplikasi Fitur Peta .....</b>	<b>62</b>
<b>Gambar 3. 25 Sequence Aplikasi Fitur <i>TalkBack</i> .....</b>	<b>63</b>
<b>Gambar 3. 26 Sequence Diagram Web Monitoring Fitur <i>Login</i>.....</b>	<b>64</b>
<b>Gambar 3. 27 Sequence Diagram Web Monitoring Fitur <i>Dashboard</i> .....</b>	<b>64</b>
<b>Gambar 3. 28 Sequence Diagram Web Monitoring Fitur Daftar Robot.....</b>	<b>65</b>

<b>Gambar 3. 29 Sequence Diagram Web Monitoring Fitur Tambah Robot.....</b>	<b>66</b>
<b>Gambar 3. 30 Sequence Diagram Web Monitoring Fitur Tambah Jenis Robot .....</b>	<b>66</b>
<b>Gambar 3. 31 Sequence Diagram Web Monitoring Fitur Daftar Jenis Robot .....</b>	<b>67</b>
<b>Gambar 4. 1 Arsitektur sistem TIFA.....</b>	<b>84</b>
<b>Gambar 4. 2 Alur Kerja Sistem Aplikasi.....</b>	<b>86</b>
<b>Gambar 4. 3 Postman insert_order.php .....</b>	<b>88</b>
<b>Gambar 4. 4 Postman get_coordinates.php.....</b>	<b>90</b>
<b>Gambar 4. 5 Postman latest-orders.php .....</b>	<b>92</b>
<b>Gambar 4. 6 Postman deliver.php.....</b>	<b>93</b>
<b>Gambar 4. 7 Tabel database orders .....</b>	<b>94</b>
<b>Gambar 4. 8 Tabel database koor .....</b>	<b>94</b>
<b>Gambar 4. 9 Skema Web Monitoring.....</b>	<b>114</b>
<b>Gambar 4. 10 Tabel Database battery_data.....</b>	<b>118</b>
<b>Gambar 4. 11 Tabel Database images.....</b>	<b>119</b>
<b>Gambar 4. 12 Tabel Database jenis_robot .....</b>	<b>119</b>
<b>Gambar 4. 13 Tabel Database orders .....</b>	<b>120</b>
<b>Gambar 4. 14 Tabel Database users.....</b>	<b>120</b>
<b>Gambar 4. 15 Tabel Database log_history .....</b>	<b>120</b>
<b>Gambar 4. 16 Tabel Database notification_history.....</b>	<b>121</b>
<b>Gambar 4. 17 IR high .....</b>	<b>145</b>
<b>Gambar 4. 18 IR low.....</b>	<b>145</b>
<b>Gambar 4. 19 Halaman Utama.....</b>	<b>146</b>
<b>Gambar 4. 20 Halaman Utama.....</b>	<b>147</b>
<b>Gambar 4. 21 Halaman Antar .....</b>	<b>147</b>
<b>Gambar 4. 22 Pop up Halaman Antar .....</b>	<b>148</b>
<b>Gambar 4. 23 Pop-up Tambah Meja.....</b>	<b>148</b>
<b>Gambar 4. 24 Pop-up Hapus Meja .....</b>	<b>149</b>
<b>Gambar 4. 25 Pop-up Konfirmasi.....</b>	<b>149</b>
<b>Gambar 4. 26 Halaman Utama.....</b>	<b>150</b>
<b>Gambar 4. 27 Halaman Media .....</b>	<b>150</b>
<b>Gambar 4. 28 Penyimpanan Media.....</b>	<b>151</b>
<b>Gambar 4. 29 Halaman Utama.....</b>	<b>151</b>
<b>Gambar 4. 30 Halaman Utama.....</b>	<b>152</b>
<b>Gambar 4. 31 Halaman Login .....</b>	<b>152</b>

<b>Gambar 4. 32 Halaman <i>Dashboard</i> .....</b>	<b>153</b>
<b>Gambar 4. 33 Halaman Daftar Robot.....</b>	<b>153</b>
<b>Gambar 4. 34 Halaman Detail Robot (1).....</b>	<b>154</b>
<b>Gambar 4. 35 Halaman Detail Robot (2) .....</b>	<b>154</b>
<b>Gambar 4. 36 <i>Pop-up</i> Log Aktivitas Robot .....</b>	<b>155</b>
<b>Gambar 4. 37 <i>Pop-up</i> Log Aktivitas Sensor .....</b>	<b>155</b>
<b>Gambar 4. 38 Halaman Metrik .....</b>	<b>156</b>
<b>Gambar 4. 39 <i>Pop-up</i> Edit Robot .....</b>	<b>156</b>
<b>Gambar 4. 40 <i>Pop-up</i> Konfirmasi Penghapusan Robot .....</b>	<b>157</b>
<b>Gambar 4. 41 Halaman Tambah Robot .....</b>	<b>157</b>
<b>Gambar 4. 42 Halaman Tambah Jenis Robot.....</b>	<b>158</b>
<b>Gambar 4. 43 Halaman Daftar Jenis Robot.....</b>	<b>159</b>
<b>Gambar 4. 44 <i>Pop-up</i> Konfirmasi Penghapusan Jenis Robot .....</b>	<b>159</b>
<b>Gambar 4. 45 <i>Pop-up</i> Notifikasi .....</b>	<b>160</b>
<b>Gambar 4. 46 Halaman Riwayat Notifikasi .....</b>	<b>160</b>
<b>Gambar 4. 47 Tombol <i>Logout</i>.....</b>	<b>161</b>
<b>Gambar 4. 48 Penyerahan Produk kepada Mitra .....</b>	<b>161</b>
<b>Gambar 5. 1 Proses Pengujian pada Barista.....</b>	<b>203</b>
<b>Gambar 5. 2 Proses Pengujian oleh Administrator.....</b>	<b>206</b>
<b>Gambar 5. 3 <i>Pie Chart</i> Analisa UAT Aplikasi.....</b>	<b>216</b>
<b>Gambar 5. 4 <i>Pie Chart</i> Analisa UAT Web Monitoring .....</b>	<b>218</b>

## DAFTAR TABEL

Tabel 3. 1 Perbandingan <i>Framework Mobile Development</i> .....	17
Tabel 3. 2 Perbandingan Database.....	18
Tabel 3. 3 Perbandingan Mikrokontroller .....	20
Tabel 3. 4 Perbandingan <i>Backend Framework</i> .....	21
Tabel 3. 5 Perbandingan Sensor Pendekksi Makanan .....	22
Tabel 3. 6 Perbandingan <i>Device</i> .....	24
Tabel 3. 7 Perbandingan <i>Framework Frontend Web</i> .....	25
Tabel 3. 8 <i>Matrix Scoring Mobile Development Framework</i> .....	30
Tabel 3. 9 <i>Matrix Scoring Database</i> .....	31
Tabel 3. 10 <i>Matrix Scoring Mikrokontroller</i> .....	32
Tabel 3. 11 <i>Matrix Scoring Backend Framework</i> .....	33
Tabel 3. 12 <i>Matrix Scoring Device</i> .....	34
Tabel 3. 13 <i>Matrix Scoring Device</i> .....	35
Tabel 3. 14 <i>Matrix Scoring Web Development Framework</i> .....	36
Tabel 3.15 <i>Matrix Scoring Sensor Tegangan</i> .....	37
Tabel 3.16 Desain UI/UX.....	67
Tabel 3. 17 Jadwal Pengerjaan .....	78
Tabel 3. 18 Anggaran.....	81
Tabel 4. 1 Tabel Penggunaan <i>Hardware</i> Aplikasi.....	113
Tabel 4. 2 Penggunaan <i>Hardware</i> Web Monitoring .....	143
Tabel 5. 1 Hasil Pengujian <i>Security Testing</i> Web Monitoring .....	165
Tabel 5. 2 Skenario Detil <i>Integration Testing</i> Aplikasi .....	166
Tabel 5. 3 Proses <i>Integration Testing</i> Aplikasi .....	167
Tabel 5. 4 Hasil <i>Integration Testing</i> Aplikasi.....	169
Tabel 5. 5 Skenario Detil <i>Integration Testing</i> Web Monitoring.....	171
Tabel 5. 6 Proses <i>Integration Testing</i> Web Monitoring .....	172
Tabel 5. 7 Hasil <i>Integration Testing</i> Web Monitoring .....	174
Tabel 5. 8 Hasil <i>Integration Testing</i> IR Sensor .....	178
Tabel 5. 9 Skenario <i>Alpha Testing</i> Aplikasi Mode Antar .....	182
Tabel 5. 10 Skenario Pengolahan Nomor Meja.....	182
Tabel 5. 11 Skenario Pengolahan Media.....	183
Tabel 5. 12 Skenario Fitur Peta .....	183

<b>Tabel 5. 13 Skenario Fitur <i>TalkBack</i>.....</b>	<b>184</b>
<b>Tabel 5. 14 Hasil Pengujian <i>Alpha Testing</i> Mode Antar .....</b>	<b>185</b>
<b>Tabel 5. 15 Hasil Pengujian <i>Alpha Testing</i> Pengolahan Nomor Meja .....</b>	<b>186</b>
<b>Tabel 5. 16 Hasil Pengujian <i>Alpha Testing</i> Pengolahan Media .....</b>	<b>186</b>
<b>Tabel 5. 17 Hasil Pengujian <i>Alpha Testing</i> Fitur Peta.....</b>	<b>187</b>
<b>Tabel 5. 18 Hasil Pengujian <i>Alpha Testing</i> Fitur <i>TalkBack</i> .....</b>	<b>187</b>
<b>Tabel 5. 19 Skenario <i>Alpha Testing</i> Web Monitoring Halaman <i>Login</i> .....</b>	<b>188</b>
<b>Tabel 5. 20 Skenario <i>Alpha Testing</i> Web Monitoring Elemen <i>Header</i>.....</b>	<b>188</b>
<b>Tabel 5. 21 Skenario <i>Alpha Testing</i> Web Monitoring Elemen <i>Sidebar</i> .....</b>	<b>189</b>
<b>Tabel 5. 22 Skenario <i>Alpha Testing</i> Web Monitoring Halaman <i>Dashboard</i>.....</b>	<b>189</b>
<b>Tabel 5. 23 Skenario <i>Alpha Testing</i> Web Monitoring Halaman Daftar Robot .....</b>	<b>191</b>
<b>Tabel 5. 24 Skenario <i>Alpha Testing</i> Web Monitoring Halaman Tambah Robot.....</b>	<b>192</b>
<b>Tabel 5. 25 Skenario <i>Alpha Testing</i> Web Monitoring Halaman Tambah Jenis Robot .</b>	<b>193</b>
<b>Tabel 5. 26 Hasil Pengujian <i>Alpha Testing</i> Halaman <i>Login</i>.....</b>	<b>194</b>
<b>Tabel 5. 27 Hasil Pengujian <i>Alpha Testing</i> Elemen <i>Header</i>.....</b>	<b>195</b>
<b>Tabel 5. 28 Hasil Pengujian <i>Alpha Testing</i> Elemen <i>Sidebar</i> .....</b>	<b>196</b>
<b>Tabel 5. 29 Hasil Pengujian <i>Alpha Testing</i> Halaman <i>Dashboard</i>.....</b>	<b>196</b>
<b>Tabel 5. 30 Hasil Pengujian <i>Alpha Testing</i> Halaman Daftar Robot.....</b>	<b>198</b>
<b>Tabel 5. 31 Hasil Pengujian <i>Alpha Testing</i> Halaman Tambah Robot.....</b>	<b>200</b>
<b>Tabel 5. 32 Hasil Pengujian <i>Alpha Testing</i> Halaman Tambah Jenis Robot.....</b>	<b>200</b>
<b>Tabel 5. 33 Bobot Nilai Setiap Kategori.....</b>	<b>201</b>
<b>Tabel 5. 34 Skenario UAT Aplikasi.....</b>	<b>202</b>
<b>Tabel 5. 35 Hasil UAT Aplikasi .....</b>	<b>203</b>
<b>Tabel 5. 36 Skenario UAT Web Monitoring .....</b>	<b>205</b>
<b>Tabel 5. 37 Hasil UAT Web Monitoring .....</b>	<b>207</b>
<b>Tabel 5. 38 Analisa Hasil UAT Aplikasi .....</b>	<b>215</b>
<b>Tabel 5. 39 Analisa Hasil UAT Web Monitoring .....</b>	<b>217</b>

## **DAFTAR SINGKATAN**

API	: Application Programming Interface
CRUD	: Create, Read, Update, Delete
DFD	: Data Flow Diagram
DRF	: Django Rest Framework
GPIO	: General Purpose <i>Input Output</i>
IDE	: Integrated Development Environment
IoT	: Internet of Things
IR	: Infrared
JSON	: JavaScript Object Notation
LiDAR	: Light Detection and Ranging
MQTT	: Message Queuing Telemetry Transport
NoSQL	: Not Only SQL
PHP	: Hypertext Preprocessor
POS	: Point of Sale
RAM	: Random Access Memory
SSD	: Solid State Drive
TIFA	: Tel-U Interactive Food Assistant
TTS	: Text to Speech
UART	: Universal Asynchronous Receiver Transmitter
UI	: <i>User</i> Interface
UAT	: <i>User Acceptance Testing</i>
UML	: Unified Modeling Language

Wi-Fi : Wireless Fidelity

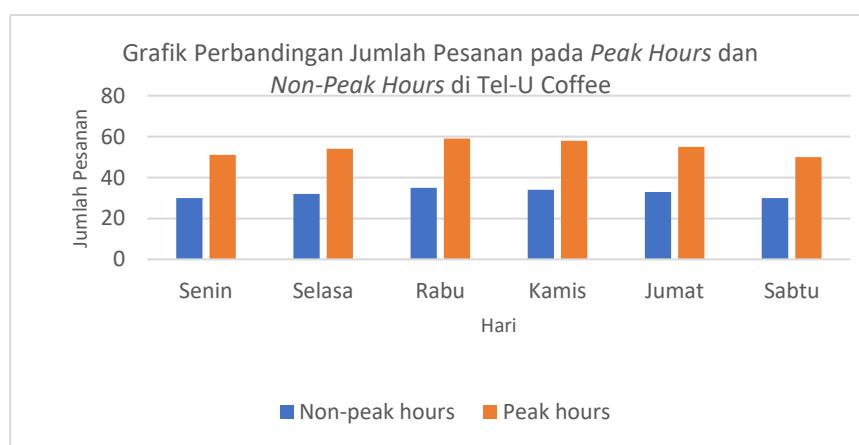
# BAB 1

## USULAN GAGASAN

### 1.1 Deskripsi Umum Masalah dan Kebutuhan

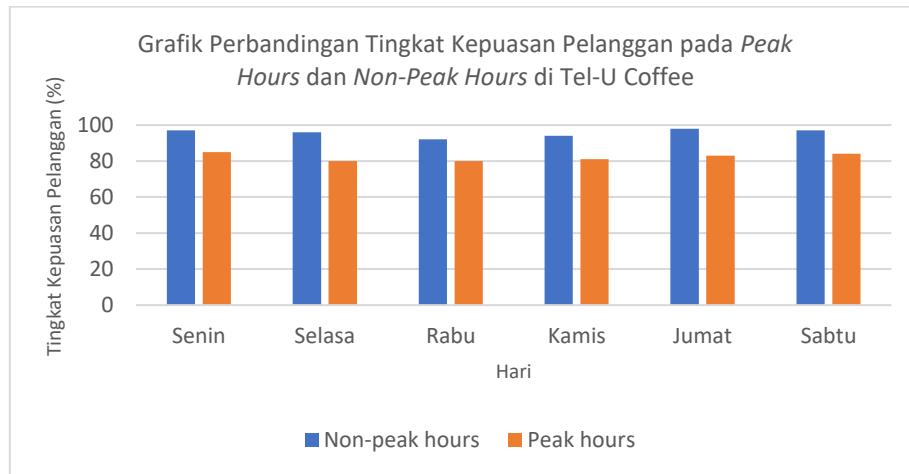
Secara global, industri kafe dan restoran menghadapi tantangan signifikan dalam menangani peningkatan permintaan pelanggan, terutama selama jam-jam sibuk (*peak hours*). Di berbagai belahan dunia, permasalahan peningkatan permintaan ini dipengaruhi oleh pertumbuhan urbanisasi, perubahan gaya hidup, serta kebutuhan akan layanan cepat dan efisien. Restoran dan kafe yang tidak mampu menangani lonjakan permintaan tersebut sering kali mengalami penurunan kualitas pelayanan, yang berdampak langsung pada kepuasan pelanggan.

Di Indonesia, khususnya di lingkungan kampus seperti Universitas Telkom, pola konsumsi mahasiswa dan staf mengalami peningkatan terutama pada waktu-waktu tertentu, seperti pagi hari sebelum jam kuliah dan siang hari saat jam makan siang. Fenomena ini juga berlaku di Tel-U Coffee, yang menjadi salah satu pusat kegiatan sosial dan akademik di lingkungan kampus. Berdasarkan survei internal yang dilakukan di Tel-U Coffee, tercatat peningkatan jumlah pesanan hingga 40% selama *peak hours* (12.00-15.00 WIB) dibandingkan waktu-waktu normal yang rata-rata pada jam sibuk jumlah pesanan meningkat sampai 50-an, terutama di hari rabu dan kamis jumlah pesanan meningkat di hari tersebut di setiap minggunya. Gambar 1.1 berikut menggambarkan perbandingan rata-rata jumlah pesanan pada *peak hours* dan non-*peak hours* di Tel-U Coffee.



**Gambar 1. 1 Grafik Perbandingan Jumlah Pesanan pada *Peak hours* dan *Non-Peak hours* di Tel-U Coffee**

Selain itu, tingkat kepuasan pelanggan juga bervariasi berdasarkan waktu pelayanan. Pada *peak hours*, keluhan pelanggan meningkat, terutama terkait keterlambatan pengiriman pesanan, ketidak tepatan pesanan, serta pelanggan yang mengeluh kepanasan akibat menumpuknya banyak pelanggan pada satu waktu. Berdasarkan data dari survei kepuasan pelanggan, tercatat penurunan tingkat kepuasan sebesar 15% pada jam-jam sibuk. Gambar 1.2, grafik yang menunjukkan perbandingan tingkat kepuasan pelanggan antara jam sibuk (*peak hours*) dan waktu-waktu normal:



**Gambar 1. 2 Grafik Perbandingan Tingkat Kepuasan Pelanggan pada *Peak hours* dan *Non-Peak hours* di Tel-U Coffee**

Manajemen Tel-U Coffee berfokus pada peningkatan efisiensi dan profitabilitas, sementara staf operasional memerlukan sistem yang mudah digunakan dan tidak mengganggu proses kerja sehari-hari, mengingat jumlah staf yang bertugas setiap harinya hanya 2 atau 3 orang sehingga para staf merasa kerepotan pada saat *peak hours*. Di sisi lain, pelanggan mengharapkan pengalaman pelayanan yang cepat dan modern. Harapan ini semakin mendesak ketika volume pesanan meningkat selama *peak hours*, sehingga kafe harus menemukan solusi yang tepat untuk menangani permasalahan ini.

Sistem pelayanan manual yang ada saat ini di Tel-U Coffee menghadapi tantangan besar dalam menangani peningkatan volume pesanan selama jam-jam sibuk. Proses pengambilan pesanan, pengolahan, hingga pengantaran sering kali tidak berjalan efisien. Selain itu, keterbatasan jumlah tenaga kerja juga menjadi salah satu faktor penyebab terjadinya *bottleneck* dalam layanan. Dengan banyaknya pesanan selama *peak hours* dan tekanan yang dihadapi staf, pelanggan kerap kali mengalami keterlambatan pesanan, kesalahan dalam pemrosesan pesanan, serta penurunan kepuasan secara keseluruhan. Tantangan ini menunjukkan bahwa

peningkatan volume pesanan tidak hanya berpengaruh pada efisiensi operasional, tetapi juga dapat memengaruhi reputasi kafe secara keseluruhan.

Penelitian yang dilakukan di Restaurant XYZ menunjukkan bahwa masalah waktu tunggu selama *peak hours* dapat berdampak signifikan pada kepuasan pelanggan. Dengan penerapan sistem pemesanan digital dan redistribusi tugas staf, waktu tunggu berhasil dikurangi dari 34 menit menjadi 24 menit, mendekati target ideal pelayanan [1]. Di industri restoran cepat saji di Indonesia, penelitian juga menunjukkan bahwa kualitas layanan dan citra merek adalah faktor utama yang memengaruhi kepuasan pelanggan. Semakin baik kualitas layanan, semakin besar kemungkinan pelanggan akan kembali dan merekomendasikan restoran kepada orang lain [2]. Selain itu, penelitian di Pizza Hut Indonesia menunjukkan bahwa kualitas layanan secara langsung mempengaruhi loyalitas pelanggan yang menunjukkan pengaruh signifikan [3].

Untuk mengatasi tantangan global dalam industri kafe dan restoran, khususnya terkait peningkatan permintaan selama jam sibuk, beberapa kafe, termasuk Tel-U Coffee, telah mencoba mengadopsi teknologi robot sebagai solusi. Robot generasi sebelumnya di Tel-U Coffee dirancang untuk membantu mengurangi beban kerja staf dan mempercepat proses pengantaran pesanan. Namun, meskipun robot ini menawarkan potensi untuk meningkatkan efisiensi, sistem yang digunakan masih memiliki sejumlah permasalahan yang membatasi efektivitasnya dalam menangani lonjakan permintaan.

Robot generasi sebelumnya di Tel-U Coffee menggunakan antarmuka pengguna yang dikembangkan dengan platform App Inventor, sebuah platform yang lebih cocok untuk aplikasi sederhana. *user interface* (UI) ini hanya menyediakan fungsionalitas dasar, seperti memasukkan perintah pengantaran, tetapi kurang fitur penting untuk pengelolaan yang lebih komprehensif, seperti pemantauan status robot atau umpan balik *real-time*. Staf kafe sering kali mengalami kesulitan dalam mengoperasikan robot, terutama pada jam sibuk, ketika interaksi yang cepat dan intuitif sangat diperlukan. Keterbatasan UI ini menghambat kemampuan staf untuk memaksimalkan penggunaan robot dalam mendukung operasional kafe.

Permasalahan lain yang dihadapi adalah tidak adanya sistem monitoring berbasis web. Tanpa adanya *Dashboard* atau antarmuka web, administrator tidak dapat memantau status baterai, atau kondisi operasional robot. Hal ini memaksa staf untuk melakukan pemeriksaan fisik secara berkala, yang tidak hanya memakan waktu tetapi juga mengurangi produktivitas. Jika terdapat sistem monitoring yang terintegrasi, administrator dapat dengan cepat

mengidentifikasi dan menangani masalah yang muncul, sehingga mengurangi *downtime* dan meningkatkan efisiensi.

Meskipun robot generasi sebelumnya mencoba mengatasi masalah global terkait peningkatan permintaan dengan mengotomatisasi sebagian proses pengantaran, permasalahan yang melekat pada sistemnya seperti navigasi yang tidak akurat, UI yang kurang optimal, dan ketiadaan sistem monitoring membatasi kemampuannya untuk memberikan solusi yang efektif. Oleh karena itu, pengembangan robot generasi baru yang mengatasi permasalahan ini menjadi krusial untuk memenuhi kebutuhan operasional kafe di lingkungan yang sibuk seperti Tel-U Coffee.

## 1.2 Analisis Masalah

Aspek analisis yang terdapat pada masalah dalam penelitian mencakup beberapa aspek. Aspek-aspek tersebut dijabarkan ke dalam aspek teknis, sumber daya manusia, dan ekonomi.

### 1.2.1 Aspek Teknis

Mengintegrasikan sistem robot *delivery* dengan sistem pemesanan yang sudah ada merupakan tantangan, terutama jika terjadi ketidaksinkronan antara pemesanan dan pengantaran yang dapat menyebabkan keterlambatan, sehingga berdampak pada kepuasan pelanggan. Kemampuan untuk memantau robot secara *real-time* pun masih terbatas, termasuk dalam hal memantau status baterai dan kondisi operasional robot. Selain itu, koneksi jaringan yang tidak stabil bisa mengganggu komunikasi antara robot dan sistem pusat, yang pada akhirnya menghambat proses pengantaran [4] [5].

Penggunaan teknologi navigasi pintar dan pembelajaran otomatis untuk robot pengantar di kafe menghadapi beberapa tantangan teknis. Pertama, teknologi navigasi yang menggunakan kamera dan sensor memerlukan perangkat yang berkualitas tinggi agar dapat bekerja dengan baik. Jika kualitas kamera atau sensor kurang baik, robot akan kesulitan menghindari tabrakan dan mencapai tujuannya, terutama jika kondisi pencahayaan kurang atau ada halangan fisik yang tidak terduga. Kedua, penerapan pembelajaran otomatis ini memerlukan algoritma yang rumit dan waktu pelatihan yang cukup lama agar robot bisa menyesuaikan diri dengan perubahan di lingkungan kafe. Proses pelatihan ini juga membutuhkan perangkat dan komponen yang canggih, yang mungkin tidak selalu tersedia dalam robot berukuran kecil dan hemat energi [6].

Antarmuka yang tidak *user-friendly* dapat menyebabkan kebingungan dan meningkatkan risiko kesalahan, sementara gangguan atau keterlambatan robot berpotensi menurunkan kepuasan pelanggan dan merusak pengalaman mereka [7].

### **1.2.2 Aspek Sumber Daya Manusia**

Pelatihan staf dalam mengoperasikan sistem robot membutuhkan waktu dan biaya tambahan. Beberapa staf mungkin enggan atau lambat beradaptasi dengan teknologi baru, yang bisa menjadi hambatan dalam pengoperasian sistem robot secara optimal. Penelitian menunjukkan bahwa pelatihan staf sangat penting dalam proses adopsi teknologi baru untuk memastikan operasional yang lancar dan efisien, serta mengurangi kekhawatiran akan penggantian pekerjaan oleh robot [8].

### **1.2.3 Aspek Ekonomi**

Investasi awal untuk membeli robot dan mengembangkan aplikasi manajemen membutuhkan biaya yang cukup besar. Namun, tidak ada jaminan bahwa investasi ini akan kembali dalam waktu dekat. Jika sistem tidak dioptimalkan dengan baik, biaya operasional bisa meningkat dan berpotensi menurunkan keuntungan. Akan tetapi, pengadopsian teknologi berbasis IoT dalam logistik dapat memberikan keuntungan dalam hal prediktif maintenance dan pengurangan waktu downtime, yang pada akhirnya menurunkan biaya operasional [9].

## **1.3 Analisis Solusi yang Ada**

Sistem pelayanan manual yang saat ini diterapkan di Tel-U Coffee memiliki keunggulan dalam hal kesederhanaan dan kemampuan memberikan layanan yang lebih personal. Namun, seiring dengan peningkatan volume pesanan, terutama pada jam-jam sibuk (*peak hours*), sistem ini terbukti tidak efektif. Keterbatasan jumlah staf, tingginya risiko kesalahan, serta lambatnya proses pemesanan dan pengantaran menyebabkan efisiensi operasional menurun, yang pada akhirnya berdampak negatif terhadap kepuasan pelanggan. Situasi ini mengindikasikan perlunya perbaikan sistem melalui adopsi teknologi yang lebih modern untuk menghadapi tantangan-tantangan tersebut.

Penerapan teknologi robot delivery mulai diterapkan di beberapa restoran untuk meningkatkan efisiensi pelayanan. Namun, robot komersial yang ada masih memiliki beberapa keterbatasan yang signifikan. Sebagai contoh, robot versi sebelumnya di Tel-U Coffee masih tidak efisien dalam lingkungan yang dinamis. Kapasitas beban robot juga terbatas hanya hingga 6 kg per *tray*, serta waktu layanan yang relatif singkat.

Selain itu, sistem robot yang sebelumnya digunakan di Tel-U Coffee memiliki beberapa kekurangan signifikan dalam aspek perangkat lunak. Salah satu permasalahan utama adalah UI yang dikembangkan menggunakan App Inventor, sebuah platform yang ditujukan untuk pembuatan aplikasi sederhana, hanya menyediakan fungsionalitas dasar dan kurang fitur penting untuk pengelolaan robot yang efektif. Staf kafe mengalami kesulitan dalam memasukkan perintah, memantau status robot, atau menerima umpan balik secara *real-time*, yang menghambat kemampuan mereka untuk mengoperasikan robot dengan efisien, terutama pada jam-jam sibuk ketika interaksi cepat sangat diperlukan [10].

Lebih lanjut, tidak adanya web monitoring interface turut memperburuk situasi. Tanpa adanya *Dashboard* berbasis web, tingkat baterai, atau status operasional robot. Hal ini mengharuskan mereka untuk melakukan pemeriksaan fisik secara berkala, yang tidak hanya membuang waktu tetapi juga menurunkan produktivitas. Sebaliknya, jika terdapat antarmuka web, administrator dapat memantau dan mengendalikan robot secara *real-time*, sehingga dapat menangani masalah dengan lebih cepat dan efisien.

Kekurangan sistemik ini yaitu ketergantungan pada UI yang sederhana, dan ketiadaan sistem monitoring berbasis web secara kolektif membatasi efektivitas robot sebelumnya dan berkontribusi pada tantangan operasional yang dihadapi oleh Tel-U Coffee. Mengatasi permasalahan ini menjadi krusial dalam pengembangan sistem robot delivery yang lebih tangguh dan efisien, yang mampu memenuhi kebutuhan kafe di lingkungan yang sibuk.

Selain tantangan teknis tersebut, penerapan robot juga membutuhkan investasi yang cukup besar, yang menjadi kendala tersendiri bagi bisnis kecil dan menengah. Oleh karena itu, adopsi teknologi ini memerlukan pertimbangan matang terkait biaya, fleksibilitas, dan kemampuan untuk menyesuaikan dengan kondisi spesifik, seperti tata letak dan alur kerja kafe. Namun, ketika melihat dari aspek jangka panjang, terdapat beberapa pertimbangan terkait biaya gaji dan biaya operasional staf. Gaji pegawai baru akan menjadi pengeluaran tetap bulanan yang perlu dipertimbangkan, termasuk tunjangan dan manfaat lainnya seperti asuransi kesehatan dan pelatihan. Biaya ini cenderung meningkat seiring dengan kenaikan inflasi dan kebutuhan untuk memberikan insentif agar staf tetap termotivasi. Menurut studi dari *Journal of Economic Perspectives*, biaya tenaga kerja di sektor layanan dapat menjadi salah satu komponen pengeluaran terbesar bagi bisnis kecil, termasuk kafe dan restoran [11].

## **1.4 Kesimpulan dan Ringkasan CD-1**

Hasil analisis masalah di Tel-U Coffee menunjukkan bahwa industri kafe, khususnya di lingkungan kampus, mengalami kesulitan dalam mengelola lonjakan permintaan selama jam sibuk (*peak hours*). Jumlah pesanan yang meningkat pesat dan keterbatasan sumber daya, termasuk minimnya jumlah staf, menimbulkan berbagai kendala operasional. Masalah ini berdampak pada penurunan kualitas layanan, seperti keterlambatan pengiriman, kesalahan pesanan, dan peningkatan keluhan pelanggan, yang berujung pada penurunan tingkat kepuasan sebesar 15% saat *peak hours*.

Permasalahan di Tel-U Coffee terkait peningkatan permintaan selama *peak hours* adalah masalah yang kompleks, melibatkan aspek teknis, sumber daya manusia, dan ekonomi. Solusi yang ada saat ini, seperti penggunaan robot, belum sepenuhnya mampu menjawab tantangan yang ada secara efektif. Oleh karena itu, diperlukan pendekatan yang lebih inovatif dan terintegrasi, baik dalam manajemen operasional maupun pengembangan teknologi, untuk memastikan Tel-U Coffee dapat memberikan pelayanan yang optimal selama *peak hours*.

Upaya penerapan robot generasi sebelumnya sebagai solusi teknologi juga menghadapi berbagai keterbatasan signifikan. Permasalahan utama meliputi antarmuka pengguna yang dikembangkan dengan App Inventor yang hanya menyediakan fungsionalitas dasar, ketiadaan sistem monitoring berbasis web untuk pemantauan *real-time*. Kekurangan-kekurangan ini membatasi efektivitas robot dalam memberikan solusi yang optimal untuk menangani lonjakan permintaan.

Permasalahan di Tel-U Coffee terkait peningkatan permintaan selama *peak hours* adalah masalah yang kompleks, melibatkan aspek teknis, sumber daya manusia, dan ekonomi. Dari aspek teknis, tantangan integrasi sistem, keterbatasan monitoring *real-time*, dan ketidakstabilan koneksi jaringan menjadi hambatan utama. Aspek sumber daya manusia menuntut investasi waktu dan biaya untuk pelatihan staf dalam mengadopsi teknologi baru. Sementara dari segi ekonomi, meskipun investasi awal teknologi robot memerlukan biaya besar, dalam jangka panjang perlu dipertimbangkan efisiensi biaya operasional dibandingkan dengan biaya tenaga kerja yang terus meningkat.

Solusi yang ada saat ini, baik sistem manual maupun penggunaan robot generasi sebelumnya, belum sepenuhnya mampu menjawab tantangan yang ada secara efektif. Oleh karena itu, diperlukan pendekatan yang lebih inovatif dan terintegrasi, baik dalam manajemen operasional maupun pengembangan teknologi yang mengatasi kekurangan sistem sebelumnya,

untuk memastikan Tel-U Coffee dapat memberikan pelayanan yang optimal selama *peak hours* dengan tetap mempertahankan kepuasan pelanggan dan efisiensi operasional.

## BAB 2

### TINJAUAN PUSTAKA

#### 2.1 Dasar Penentuan Spesifikasi

Perencanaan spesifikasi robot pengantar makanan dan aplikasi monitoring dilakukan dengan menetapkan batasan dan spesifikasi yang relevan sebelum pengembangan dimulai, perlu dicatat bahwa pengembangan robot pengantar makanan ini dikerjakan oleh tiga tim yang berasal dari berbagai program studi. Tim pertama, fokus pada kontrol, kinematik, mekanik, dan power, berasal dari Program Studi S1 Teknik Elektro. Tim kedua, bertanggung jawab untuk bagian mapping, path planning, dan obstacle avoidance menggunakan Motion ROS, berasal dari Program Studi S1 Teknik Telekomunikasi. Sementara itu, tim kami dari Program Studi S1 Teknik Komputer, berfokus pada pengembangan IoT *Software* dan aplikasi monitoring robot, termasuk pembuatan aplikasi serta pengelolaan database. Setiap tim memiliki peran penting dalam menyelesaikan bagian spesifik dari sistem, bekerja secara sinergis untuk menghasilkan robot pengantar makanan yang dapat beroperasi dengan efisien dan handal. Spesifikasi ini disusun berdasarkan referensi dari berbagai sumber, termasuk produk serupa (seperti Bellabot dari Pudu Robotics), robot pengantar sebelumnya di Tel-U Coffee, serta standar pelayanan operasional yang berlaku ini [10] [12].

Teknologi robot pengantar makanan telah berkembang pesat, salah satunya adalah Bellabot, yang diimplementasikan secara komersial. Bellabot dilengkapi fitur utama seperti pemantauan status baterai secara *real-time*, visualisasi peta robot, rute tujuan, serta pengaturan *tray*, dan mode operasional [12]. Fitur-fitur ini menjadi dasar pengembangan sistem monitoring dan manajemen robot di Tel-U Coffee, dengan penyesuaian untuk kebutuhan.

Selain itu, pengalaman pengembangan robot pengantar sebelumnya di Tel-U Coffee memberikan dasar untuk fitur seperti "Obstacle Avoidance" untuk penghindaran objek otomatis, serta "Adjustable Tray" yang dapat menampung hingga 6 kg per *tray*. Fitur tambahan seperti "Sliding Door" memastikan keamanan dan kebersihan selama pengantaran [10] [13] [14].

Robot pengantar sebelumnya dan standar pelayanan di Tel-U Coffee juga menjadi acuan untuk desain sistem, di mana pelanggan dapat langsung menerima pesanan di meja mereka tanpa perlu ke area pengambilan. Dengan fitur seperti sistem antrean meja yang mendukung hingga tiga nomor meja dalam satu kali pengantaran, robot ini dirancang untuk efisiensi

operasional selama jam sibuk. Untuk mendukung kelancaran operasional, aplikasi monitoring dari *Device* memudahkan barista dan pelanggan dalam mengakses informasi pesanan [15].

Pengembangan aplikasi pada robot dan web monitoring diperlukan untuk mengoptimalkan operasional robot pengantar makanan di Tel-U Coffee. Aplikasi dirancang sebagai penghubung utama antara penyedia layanan dan pengguna, dilengkapi dengan berbagai mode operasional yang fleksibel untuk memenuhi kebutuhan yang berbeda. Selain itu, web monitoring disediakan sebagai alat untuk memantau aktivitas robot secara *real-time*.

Aplikasi membantu barista dalam pengantaran makanan dan minuman ke meja pelanggan, terutama saat jam sibuk. Aplikasi dirancang dengan antarmuka yang intuitif, menawarkan berbagai tombol utama yang memungkinkan pengguna untuk mengakses fitur utama, melakukan konfigurasi sesuai kebutuhan, dan memulai proses yang diinginkan. Setelah semua preferensi ditetapkan dan perintah untuk memulai diberikan, aplikasi secara otomatis melanjutkan ke tahap operasional yang relevan, memastikan kelancaran proses sesuai dengan fungsi yang telah diatur.

Sistem ini mendukung interaksi langsung yang memungkinkan pengguna untuk menyampaikan kebutuhan secara *real-time* selama operasional berlangsung. Selain itu, saat proses mencapai titik tujuan, pengguna akan menerima pemberitahuan yang relevan, dilengkapi dengan opsi untuk mengonfirmasi penyelesaian setelah kebutuhan terpenuhi.

Web monitoring digunakan untuk memantau robot secara *real-time*. Web ini dirancang untuk komunikasi data yang dilakukan melalui protokol Node.JS [16].

Desain antarmuka aplikasi menerapkan tema modern untuk menciptakan kesan minimalis dan profesional, serta memastikan kemudahan penggunaan bagi barista maupun pelanggan. Fokus utama pengembangan adalah pada peningkatan dari App Inventor sebelumnya yang dinilai memiliki antarmuka yang kurang menarik dan kurang intuitif.

Tantangan utama dalam pengembangan aplikasi ini adalah meningkatkan pengalaman pengguna (UI/UX) dari aplikasi sebelumnya yang memiliki desain antarmuka yang tidak menarik. Untuk mengatasi tantangan ini, tim pengembang berfokus pada pembuatan desain yang lebih modern dan interaktif, sekaligus mempertahankan fungsionalitas inti yang sudah ada. Strategi ini bertujuan untuk meningkatkan daya tarik aplikasi tanpa mengorbankan kemudahan operasional [10].

## **2.2 Daftar Batasan dan Spesifikasi**

Berikut penjelasan mengenai daftar batasan dan spesifikasi robot pengantar makanan beserta sistem monitoring yang dirancang untuk memenuhi standar operasional di Tel-U Coffee:

### **2.2.1 Daftar Batasan dan Spesifikasi Robot**

Robot pengantar makanan ini dirancang dengan tampilan antarmuka pengguna (*User Interface*) yang menarik dan intuitif, memudahkan interaksi antara operator dan robot. Dengan dukungan *Hardware* seperti prosesor yang kompatibel dengan komunikasi nirkabel (Wi-Fi), fitur ini dapat diakses melalui perangkat *mobile*. Long Service Time memastikan robot dapat beroperasi dalam durasi panjang, cocok untuk kebutuhan layanan pada jam sibuk, dengan motor servo yang mampu menopang beban hingga 10 kilogram.

Selain itu, robot dilengkapi dengan Adjustable *Tray*, memungkinkan penyesuaian *tray* untuk berbagai jenis pesanan. Mekanisme ini menggunakan aktuator linear dengan panjang stroke minimal 100 mm, memberikan fleksibilitas dalam pengantaran barang. Fitur Sliding Door memberikan perlindungan tambahan pada makanan atau minuman yang diantarkan, menjaga keamanan dan kebersihan selama proses pengiriman. Dengan spesifikasi ini, robot pengantar makanan dirancang untuk memberikan layanan yang optimal dan memenuhi kebutuhan operasional secara efisien [17].

### **2.2.2 Daftar Batasan dan Spesifikasi Aplikasi dan Web Monitoring**

Aplikasi dirancang untuk memfasilitasi pengawasan dan pengendalian robot pengantar makanan di Tel-U Coffee. Visualisasi posisi robot ditampilkan dalam bentuk real-world mapping yang memanfaatkan hasil proses Mapping, di mana setiap koordinat meja dimasukkan ke dalam tabel pilihan meja pada halaman utama aplikasi [18]. Untuk menjalankan fitur ini, minimum perangkat yang dibutuhkan adalah *Device* dengan prosesor octa-core, RAM 3 GB, penyimpanan internal 32 GB, serta layar beresolusi 1920 x 1080. Dengan kontrol yang intuitif, aplikasi mendukung tiga mode operasional, yaitu “mode Antar” untuk pengantaran pesanan, “fitur Media”, “fitur Peta”, dan “fitur *TalkBack*”.

Selain itu, aplikasi mendukung pengisian nomor pesanan yang terhubung langsung dengan database pemesanan kafe, dilengkapi validasi oleh barista untuk memastikan *input* nomor pesanan sesuai [19].

Monitoring baterai robot dilakukan secara *real-time* dengan indikator persentase yang tampil pada aplikasi, disertai *Pop-up* notification otomatis pada web ketika daya mendekati

kapasitas rendah [12]. Teknologi yang digunakan dalam aplikasi membutuhkan platform pengembangan perangkat lunak dengan dukungan *debugging real-time* dan kemampuan untuk membangun aplikasi secara efisien. Komunikasi data antara aplikasi dan robot memerlukan protokol komunikasi yang mendukung pengiriman data secara cepat dan akurat dengan latensi rendah untuk memastikan performa *real-time* [16]. Sistem ini juga terhubung dengan basis data yang memiliki kemampuan menyimpan dan mengelola data historis untuk keperluan monitoring dan controlling [20].

Aplikasi dirancang dengan antarmuka pengguna bertema modern, serta mendukung penggunaan dalam Bahasa Indonesia. Fitur *Pop-up notification* tersedia untuk memberi tahu operator tentang status robot, seperti ketika robot tiba di meja tujuan. Sistem saat ini mendukung pengelolaan satu robot dan belum terintegrasi langsung dengan sistem POS atau mendukung ekspansi *multi-robot*. Meski begitu, aplikasi ini telah dirancang untuk memenuhi kebutuhan operasional kafe secara optimal.

Web monitoring dirancang untuk memfasilitasi pengawasan sistem robotik secara *real-time*. Sistem ini menampilkan informasi penting seperti status robot, persentase baterai, dan log aktivitas sistem. Minimum untuk server mencakup 1 vCPU, RAM 1 GB, penyimpanan SSD 25 GB, dan sistem operasi Ubuntu 22.04 LTS. Server juga harus mendukung proses deployment CI/CD, konfigurasi firewall (UFW), serta pengamanan menggunakan SSL. Web monitoring juga harus kompatibel dengan berbagai ukuran layar dan *browser* umum, serta mudah dioperasikan dengan pelatihan minimal [21]. Aspek legal dan keamanan juga menjadi prioritas, seperti akses sistem yang hanya diberikan kepada pengguna yang terotorisasi [22]. Click or tap here to enter text.

### 2.2.3 Batasan Sistem

Sistem robot pengantar makanan beserta platform monitoring di Tel-U Coffee memiliki sejumlah batasan yang penting diperhatikan dalam pengembangannya. Pertama, robot ini hanya dirancang untuk beroperasi di dalam ruangan yang telah dipetakan sebelumnya di Tel-U Coffee. Robot ini tidak dioptimalkan untuk penggunaan di luar ruangan atau area dengan medan yang tidak rata atau belum dipetakan, karena "sistem robot pengantar makanan sering kali dirancang untuk penggunaan di dalam ruangan yang telah dipetakan sebelumnya, dengan tujuan mengoptimalkan efisiensi operasional dalam lingkungan terkendali" [18].

Selain itu, robot memiliki batasan dalam hal jumlah meja tujuan yang dapat dilayani, yaitu maksimal hanya tiga meja dalam satu kali pengantaran. Sistem ini memungkinkan

pengantar beberapa pesanan secara bersamaan, namun untuk menjaga efisiensi dan akurasi, jumlah meja tujuan tidak boleh melebihi batas tersebut. Kapasitas beban setiap *tray* juga terbatas hingga maksimal 10 kilogram per *tray*, sehingga barang yang dibawa harus berada dalam kapasitas tersebut atau kurang untuk menjaga stabilitas dan keamanan pengantaran. Sebagaimana dijelaskan, "*tray* robot pengantar makanan umumnya memiliki batas kapasitas beban hingga 10 kilogram, sesuai standar operasional untuk menjaga stabilitas selama pengantaran" [19].

Untuk menghindari tabrakan dengan rintangan, sistem Obstacle Avoidance pada robot telah dirancang untuk mendeteksi dan menghindari rintangan statis maupun dinamis yang sudah dikenali. Namun, kemampuan deteksi terhadap rintangan yang sangat kecil atau transparan mungkin tidak optimal, sebagaimana disebutkan bahwa "kemampuan penghindaran rintangan robot sangat bergantung pada sensor yang digunakan, namun masih memiliki keterbatasan dalam mendeteksi objek kecil atau transparan" [23].

Sistem robot pengantar memungkinkan penggunaan *tray* yang dapat disesuaikan, satu pesanan dapat dimuat dengan beberapa *tray* (One To Many), akan tetapi satu *tray* hanya dapat memuat satu pesanan (One To One). Fleksibilitas *tray* ini sesuai dengan pernyataan bahwa "penggunaan *tray* yang dapat disesuaikan telah menjadi "fitur Antar" pada banyak robot pengantar modern, memungkinkan pengangkutan beberapa item dalam sekali pengantaran" [10].

Dari sisi konektivitas, web monitoring dirancang untuk beroperasi diluar area Tel-U Coffee. Sistem ini mendukung akses jarak jauh melalui internet. Hal ini sejalan dengan pandangan bahwa "penggunaan jaringan untuk komunikasi antara web monitoring dan robot membantu menjaga kecepatan dan stabilitas sistem dibandingkan akses berbasis internet" [12]. Batasan-batasan ini disusun untuk mempertimbangkan aspek keamanan, efisiensi, dan keandalan sistem robot pengantar makanan, agar sistem dapat berfungsi secara optimal sesuai skenario layanan di Tel-U Coffee.

### 2.3 Pengukuran/Verifikasi Spesifikasi

Dalam pembuatan aplikasi dan web monitoring ini diperlukan pengujian, pengujian yang diimplementasikan pada aplikasi dan web monitoring menggunakan metode *User Acceptance Testing* (UAT). Dalam pengujian aplikasi monitoring dan pengendalian robot pengantar

makanan di Tel-U Coffee, UAT dilakukan dengan melibatkan administrator dan barista. Pengujian mencakup skenario utama seperti pemilihan posisi *tray* dan meja tujuan untuk pengantaran serta validasinya oleh barista, pengaturan mode kerja, serta monitoring baterai robot secara *real-time*. Selain itu, fitur notifikasi *Pop-up* diuji untuk memastikan pesan muncul sesuai kondisi, seperti robot tiba di meja tujuan. UAT juga menguji kegunaan aplikasi dengan memantau bagaimana pengguna menggunakan fitur aplikasi untuk mengkonfirmasi pesanan selesai dalam situasi operasional nyata termasuk saat jam sibuk. Semua hasil pengujian, termasuk umpan balik pengguna dan catatan error, didokumentasikan untuk evaluasi lebih lanjut. Jika aplikasi berhasil memenuhi kriteria yang telah ditetapkan, maka aplikasi dinyatakan siap untuk implementasi operasional. Namun, jika ditemukan kekurangan, pengembang akan melakukan perbaikan sebelum dilakukan pengujian ulang [24].

Pengujian web monitoring robot pengantar makanan dengan metode *User Acceptance Testing* (UAT) dilakukan untuk memastikan bahwa sistem memenuhi kebutuhan operasional pengguna akhir dan siap diimplementasikan. UAT dimulai dengan perencanaan yang menetapkan tujuan pengujian, seperti memastikan fitur utama, yaitu tampilan status *real-time* robot (baterai dan log aktivitas robot) berjalan sesuai kebutuhan. Selanjutnya, lingkungan pengujian disiapkan menggunakan jaringan dan data robot simulasi untuk menciptakan kondisi yang menyerupai penggunaan nyata. Skrip pengujian disusun dengan mencakup skenario utama, seperti pembaruan data *real-time*, fungsi multi-*login* hingga lima pengguna, dan pengujian autentikasi. Pada tahap eksekusi, pengguna menjalankan skenario tersebut dan mencatat hasil pengujian, termasuk masalah seperti waktu respon yang lambat atau ketidaksesuaian tampilan antarmuka. Setelah pengujian, hasilnya dievaluasi dengan indikator keberhasilan, seperti keberhasilan skenario  $\geq 95\%$ , tingkat kepuasan pengguna  $\geq 80\%$ , dan performa sistem yang stabil dengan pengguna simultan. Jika ditemukan masalah, pengembang melakukan perbaikan sebelum mengulang pengujian untuk memastikan semua masalah teratasi. Jika semua kriteria terpenuhi, pengguna memberikan persetujuan bahwa sistem siap digunakan. Dengan UAT, keandalan, kemudahan penggunaan, dan kesesuaian sistem terhadap kebutuhan pengguna dapat dipastikan sebelum implementasi penuh [25].

## 2.4 Kesimpulan dan Ringkasan CD-2

Penyusunan batasan dan spesifikasi sistem robot pengantar makanan beserta aplikasi monitoring dan controlling serta web monitoring untuk mendukung operasional di Tel-U Coffee. Penyusunan spesifikasi dilakukan berdasarkan referensi dari produk serupa seperti

Bellabot, pengalaman pengembangan robot sebelumnya, dan standar pelayanan di Tel-U Coffee, sehingga menghasilkan solusi yang relevan, terukur, dan *realistik*. Batasan dan spesifikasi yang telah dirumuskan mencakup berbagai aspek, termasuk fitur robot *obstacle avoidance*, dan *adjustable tray*, serta kemampuan aplikasi untuk mengatur mode robot, memantau status pesanan, dan memberikan notifikasi kepada pengguna. Di sisi lain, web monitoring dirancang untuk memantau robot secara *real-time* dengan efisiensi tinggi melalui jaringan.

Pengujian spesifikasi dilakukan menggunakan metode *User Acceptance Testing* (UAT), melibatkan operator, barista, dan staf kafe sebagai pengguna utama. Pengujian ini memastikan aplikasi dan sistem monitoring memenuhi kebutuhan operasional, seperti kemudahan penggunaan, *responsivitas*, dan stabilitas sistem, termasuk dalam skenario jam sibuk. Dengan dokumentasi hasil pengujian yang mencakup *feedback* pengguna dan perbaikan yang diperlukan, pengembangan aplikasi dan web monitoring diarahkan untuk menghasilkan sistem yang siap diimplementasikan dan sesuai ekspektasi pengguna.

Keseluruhan sistem dirancang untuk meningkatkan efisiensi layanan di Tel-U Coffee dengan mengurangi beban manual barista dan memberikan pengalaman pengguna yang optimal. Dengan pendekatan desain berbasis kebutuhan pengguna dan pengujian menyeluruh, sistem robot pengantar makanan ini diharapkan mampu memberikan solusi yang handal dan inovatif bagi operasional kafe.

## **BAB 3**

### **SPESIFIKASI DAN DESAIN SISTEM**

#### **3.1 Alternatif Usulan Solusi**

Setelah melakukan analisis terhadap kebutuhan sistem, beberapa alternatif usulan solusi telah dirumuskan untuk memastikan bahwa sistem yang dikembangkan dapat berfungsi secara optimal sesuai tujuan. Setiap komponen dari sistem ini memerlukan pendekatan spesifik berdasarkan kebutuhan teknis dan operasionalnya. Oleh karena itu, pembahasan berikutnya akan menjelaskan secara rinci solusi yang diusulkan untuk setiap komponen utama, meliputi pengembangan aplikasi, pengelolaan basis data, perangkat keras, serta kerangka kerja untuk *backend* dan *Frontend*. Rincian tersebut akan diuraikan dalam sub bab berikut.

##### **3.1.1 Usulan Solusi *Mobile Development Framework***

Kotlin untuk pengembangan aplikasi Android yang menawarkan keunggulan dalam hal integrasi mendalam dengan Android SDK dan fitur-fitur eksklusif seperti emulator untuk pengujian langsung, serta *tools debugging* yang kuat. Dengan menggunakan Android Studio, pengembang dapat memanfaatkan seluruh kemampuan sistem Android, seperti akses ke API khusus Android dan optimasi aplikasi dengan native performace. Kekurangannya, Android Studio cenderung berat dan memerlukan spesifikasi *Hardware* tinggi untuk berjalan lancar, terutama pada komputer dengan RAM terbatas. Selain itu, pengembangan aplikasi Android hanya berlaku untuk platform Android saja [26].

Flutter, dikembangkan oleh Google, memungkinkan pengembangan aplikasi lintas platform dengan satu basis kode untuk Android, iOS, web, dan desktop. Keunggulannya mencakup tampilan konsisten, performa tinggi berkat engine rendering bawaan, dan fitur hot reload yang mempercepat pengembangan. Namun, Flutter memiliki kekurangan dalam dukungan beberapa plugin native, yang kadang memerlukan implementasi kode platform khusus. [27].

React Native memungkinkan pengembangan aplikasi lintas platform menggunakan JavaScript dan React, dengan kemampuan berbagi kode untuk Android dan iOS. Keunggulannya meliputi percepatan pengembangan melalui kode bersama dan dukungan komunitas aktif. Selain itu, React Native dapat terintegrasi dengan komponen native. Selain itu, React Native dapat terintegrasi dengan komponen native untuk meningkatkan kinerja.

Namun, kekurangannya mencakup potensi masalah kinerja pada aplikasi kompleks, karena komponen harus diterjemahkan dari JavaScript ke native. Tabel 3.1 di bawah ini menyajikan perbandingan ketiga *Framework mobile* development berdasarkan kriteria tertentu [27].

**Tabel 3. 1 Perbandingan *Framework Mobile* Development**

Kriteria	Kotlin	Flutter	React Native
Penggunaan Utama	Pengembangan aplikasi Android	Pengembangan aplikasi lintas platform (Android, iOS, web, desktop)	Pengembangan aplikasi lintas platform (Android, iOS)
Bahasa Pemrograman	Java, Kotlin	Dart	JavaScript
Emulator	Emulator Android bawaan	Tidak ada emulator bawaan	Tidak ada emulator bawaan
Kemudahan Penggunaan	Membutuhkan pengalaman lebih dengan Android	Relatif mudah, tetapi memerlukan pemahaman Dart	Mudah, terutama bagi yang familiar dengan JavaScript
Cross-Platform	Tidak (khusus untuk Android)	Ya (Android, iOS, web, desktop)	Ya (Android, iOS)
Kekurangan Utama	Berat, membutuhkan spesifikasi komputer tinggi	Plugin/platform support kadang terbatas	Tidak optimal untuk aplikasi kompleks
Cocok Untuk	Aplikasi native Android	Aplikasi lintas platform dengan desain konsisten	Aplikasi lintas platform yang membutuhkan kustomisasi tinggi

### 3.1.2 Usulan Solusi Database

MySQL adalah database relasional yang populer dengan struktur tabel terorganisir dan dukungan untuk *query* SQL yang kuat. Keunggulannya adalah kemampuannya untuk

menangani data yang terstruktur dan terorganisir dengan baik seperti riwayat pengiriman, status robot, dan informasi pengguna. MySQL sangat cocok untuk aplikasi yang memerlukan operasi CRUD kompleks dan join antar tabel. Kekurangannya, MySQL mungkin kurang efisien untuk menangani data non-relasional atau skala besar dalam aplikasi *real-time* seperti robot, terutama ketika perubahan data sering terjadi [28].

Firebase adalah platform NoSQL yang menyediakan layanan *backend-as-a-service* (BaaS), termasuk Realtime Database dan Firestore, yang sangat cocok untuk aplikasi yang membutuhkan komunikasi *real-time* dan sinkronisasi data secara instan. Keunggulannya adalah kemudahan integrasi dengan aplikasi *mobile*, kemampuan *real-time* updates, dan scalability tanpa perlu pengelolaan server. Kekurangannya, Firebase memiliki skema harga berdasarkan penggunaan. Selain itu, struktur data NoSQL-nya mungkin tidak sesuai dengan kebutuhan aplikasi yang memerlukan relasi antar data yang kompleks [29].

MongoDB adalah database NoSQL yang menyimpan data dalam format dokumen JSON dan mendukung penyimpanan data non-relasional. Keunggulannya adalah kemampuannya untuk menangani data yang tidak terstruktur dan mendukung skala horizontal dengan lebih mudah. Kekurangannya adalah jika tidak diatur dengan baik, MongoDB bisa mengalami masalah dengan konsistensi data dan kompleksitas *query* yang sulit dibandingkan SQL. Pada tabel 3.2 di bawah ini dipaparkan perbandingan antara ketiga usulan solusi database berdasarkan kriteria yang telah ditentukan [30].

**Tabel 3.2 Perbandingan Database**

Kriteria	MySQL	Firebase	MongoDB
Jenis Database	Relasional	NoSQL	NoSQL
Penggunaan Utama	Menyimpan data terstruktur dan terorganisir	Sinkronisasi data <i>real-time</i> antar perangkat	Penyimpanan data tidak terstruktur
Keunggulan	Mendukung <i>query</i> SQL yang kuat dan operasi CRUD	<i>Real-time</i> updates dan skalabilitas tinggi tanpa pengelolaan server	Fleksibilitas tinggi dan mudah di-scaling secara horizontal

Kriteria	MySQL	Firebase	MongoDB
Kekurangan	Kurang efisien untuk data non-relasional atau <i>real-time</i>	Berbayar dengan skema harga berbasis penggunaan	Kompleksitas <i>query</i> lebih tinggi dibandingkan SQL
Cocok Untuk	Aplikasi dengan kebutuhan relasi data yang kompleks	Aplikasi dengan komunikasi dan sinkronisasi data <i>real-time</i>	Aplikasi dengan data yang sering berubah seperti sensor dan status robot

### 3.1.3 Usulan Solusi Mikrokontroller

ESP32 adalah mikrokontroler dengan kemampuan Wi-Fi dan Bluetooth terintegrasi, membuatnya sangat cocok untuk aplikasi *Internet of Things* (IoT) dan komunikasi nirkabel. Keunggulannya adalah memiliki prosesor dual-core, kecepatan tinggi (hingga 240 MHz), dan memori lebih besar dibandingkan dengan mikrokontroler lainnya, serta kemampuan multitasking. ESP32 juga mendukung lebih banyak antarmuka seperti GPIO, SPI, I2C, PWM, dan banyak lagi. Kekurangannya, kompleksitas pemrograman bisa lebih tinggi dibandingkan dengan mikrokontroler lain yang lebih sederhana, dan ada konsumsi daya lebih tinggi dibandingkan dengan ESP8266 atau Arduino [31].

ESP8266 adalah mikrokontroler yang lebih sederhana dibandingkan dengan ESP32, tetapi masih memiliki kemampuan Wi-Fi terintegrasi. Keunggulannya adalah harga yang lebih terjangkau dan kemudahan pemrograman menggunakan Arduino IDE, serta ukuran yang kecil dan rendah konsumsi daya jika dibandingkan dengan ESP32. ESP8266 sangat cocok untuk aplikasi yang membutuhkan konektivitas Wi-Fi namun tidak memerlukan banyak prosesor atau fitur tambahan. Kekurangannya adalah keterbatasan memori dan kecepatan prosesor yang lebih rendah dibandingkan dengan ESP32, serta tidak mendukung Bluetooth [32].

Arduino Nano adalah salah satu mikrokontroler berbasis ATmega328P yang sangat populer, terutama untuk *prototyping* dan proyek kecil. Keunggulannya adalah kemudahan penggunaan, kompatibilitas dengan Arduino IDE, dan banyaknya pustaka serta shield yang tersedia. Arduino Nano juga memiliki konsumsi daya rendah, menjadikannya ideal untuk aplikasi yang membutuhkan penghematan daya, serta sangat cocok untuk proyek non-IoT seperti kontrol motor, sensor, dan antarmuka sederhana. Kekurangannya adalah tidak memiliki kemampuan Wi-Fi atau Bluetooth bawaan, sehingga untuk aplikasi IoT, Anda harus

menambahkan modul tambahan. Pada tabel 3.3 dipaparkan perbandingan antara ketiga usulan solusi mikrokontroller dengan kriteria yang telah ditentukan [33].

**Tabel 3. 3 Perbandingan Mikrokontroller**

Kriteria	ESP32	ESP8266	Arduino Nano
Kemampuan Utama	Wi-Fi dan Bluetooth terintegrasi	Wi-Fi terintegrasi	Mikrokontroler untuk <i>prototyping</i> tanpa Wi-Fi/Bluetooth
Keunggulan	Prosesor dual-core, kecepatan tinggi, multitasking	Harga terjangkau, rendah konsumsi daya	Mudah digunakan, kompatibilitas tinggi dengan Arduino IDE
Kekurangan	Kompleksitas pemrograman tinggi, konsumsi daya lebih besar	Memori dan kecepatan prosesor terbatas	Tidak memiliki Wi-Fi/Bluetooth bawaan
Cocok Untuk	Aplikasi IoT yang kompleks dan multitasking	Aplikasi IoT sederhana dengan konektivitas Wi-Fi	Proyek non-IoT seperti kontrol motor atau sensor

### 3.1.4 Usulan Solusi *Backend Framework*

Node.js adalah runtime berbasis JavaScript yang sangat populer, terutama untuk aplikasi yang membutuhkan komunikasi *real-time*. Keunggulannya terletak pada kemampuannya untuk menangani banyak koneksi secara bersamaan melalui model *non-blocking* yang efisien, sehingga cocok untuk aplikasi seperti monitoring dan controlling robot pengantar makanan yang memerlukan pembaruan status secara instan. Node.js dapat menangani request dengan sangat cepat dan mendukung berbagai protokol komunikasi *real-time* seperti WebSocket, yang sangat berguna untuk mengontrol robot secara langsung. Di sisi lain, kekurangannya adalah pengelolaan kode asinkron yang bisa menjadi kompleks, terutama ketika proyek semakin besar dan melibatkan banyak interaksi [34].

Django adalah *Framework* Python yang terkenal karena kemudahan dan kecepatan dalam membangun aplikasi web yang aman dan skalabel. Keunggulannya terletak pada kemudahan pengelolaan database dan integrasi API, sehingga sangat baik untuk aplikasi yang

membutuhkan pengolahan data yang lebih kompleks, seperti pelaporan, autentikasi, dan manajemen data pengguna. Dengan menggunakan Django Rest Framework (DRF), Anda dapat dengan mudah membuat API RESTful untuk komunikasi antara aplikasi Android dan robot. Django lebih cocok untuk aplikasi dengan pengolahan data *backend* yang intensif, namun tidak seoptimal Node.js untuk komunikasi *real-time* [35].

Spring Boot adalah *Framework* berbasis Java yang menawarkan scalability dan keamanan tinggi, serta kemampuan untuk membangun aplikasi yang sangat stabil dan mudah untuk dikelola, terutama dalam aplikasi besar dan kompleks. Spring Boot mendukung komunikasi *real-time* melalui Spring WebSocket dan Spring Integration, yang memungkinkan aplikasi Anda untuk mengontrol robot secara langsung. Keunggulannya terletak pada sistem keamanan yang kuat, dukungan untuk microservices, dan kemampuan untuk menangani aplikasi skala besar. Namun, bagi aplikasi yang lebih sederhana dan dengan fokus pada *real-time* interaction, Spring Boot bisa terasa lebih berat dan kompleks dibandingkan dengan Node.js. Pada tabel 3.4 dipaparkan perbandingan antara ketiga usulan solusi *backend Framework* dengan kriteria yang telah ditentukan [36].

**Tabel 3. 4 Perbandingan *Backend Framework***

Kriteria	Node.js	Django	Spring Boot
Penggunaan Utama	Komunikasi <i>real-time</i> dan konektivitas server	Pengelolaan data <i>backend</i> dan API RESTful	Aplikasi skala besar dengan keamanan tinggi
Keunggulan	<i>Non-blocking</i> , mendukung protokol <i>real-time</i> seperti WebSocket dan MQTT	Mudah membuat API RESTful, integrasi database yang kuat	Dukungan microservices, keamanan kuat, dan stabilitas tinggi
Kekurangan	Kompleksitas pengelolaan kode asinkron	Kurang optimal untuk komunikasi <i>real-time</i>	Kompleksitas tinggi untuk aplikasi sederhana
Cocok Untuk	Aplikasi monitoring dan controlling dengan real-updates time	Aplikasi dengan pengolahan data <i>backend</i> intensive	Aplikasi besar yang membutuhkan stabilitas dan keamanan tinggi

### 3.1.5 Usulan Solusi Sensor Pendekripsi Makanan

Load Cell mengukur perubahan berat atau gaya yang diterima oleh sensor. Sensor ini sangat cocok untuk aplikasi yang membutuhkan deteksi keberadaan objek berdasarkan berat, seperti mengetahui apakah *tray* sudah terisi makanan. Kelebihan utama dari load cell adalah akurasi dalam mendekripsi perubahan berat, sehingga dapat memberikan informasi yang lebih spesifik tentang seberapa banyak makanan yang ada di *tray*. Namun, kekurangannya adalah sensor ini hanya memberikan informasi tentang berat, dan tidak dapat mendekripsi jenis atau posisi makanan secara detail [37].

IR Sensor (Infrared) bekerja dengan memancarkan cahaya inframerah dan mengukur jumlah cahaya yang dipantulkan kembali oleh objek. Sensor ini sangat sederhana dan efektif untuk mendekripsi keberadaan objek dalam jarak dekat tanpa kontak langsung, cocok untuk deteksi makanan yang mudah disorot dengan cahaya inframerah. Kelebihan IR sensor adalah harga yang relatif murah dan kemampuan deteksi objek yang cepat dalam jarak dekat. Namun, kekurangannya adalah sensitivitasnya terhadap cahaya sekitar yang bisa mengganggu kinerja [38].

Ultrasonic Sensor bekerja dengan mengukur jarak menggunakan gelombang suara. Sensor ini dapat digunakan untuk mendekripsi objek berdasarkan jarak atau ketinggian objek di dalam *tray*. Kelebihan dari ultrasonic sensor adalah kemampuannya untuk mendekripsi objek dengan jarak yang lebih jauh dan tanpa terpengaruh oleh cahaya. Namun, kekurangannya adalah ketepatan pengukurannya dapat terpengaruh oleh bentuk atau permukaan objek yang tidak rata, dan tidak memberikan informasi tentang jenis objek. Pada tabel 3.5 dipaparkan perbandingan antara ketiga usulan solusi sensor pendekripsi makanan dengan kriteria yang telah ditentukan [33] [38].

**Tabel 3. 5 Perbandingan Sensor Pendekripsi Makanan**

Kriteria	Load Cell	IR Sensor	Ultrasonic Sensor
Fungsi Utama	Mengukur perubahan berat atau gaya	Mendekripsi keberadaan objek melalui cahaya inframerah	Mengukur jarak menggunakan gelombang suara

Kriteria	Load Cell	IR Sensor	Ultrasonic Sensor
Keunggulan	Akurat mendeteksi perubahan berat	Murah, cepat untuk deteksi objek jarak dekat	Dapat mendeteksi objek jarak jauh, tidak terpengaruh cahaya
Kekurangan	Hanya mendeteksi berat	Sensitif terhadap cahaya sekitar	Kurang akurat pada permukaan objek yang tidak rata
Cocok Untuk	Deteksi keberadaan objek berdasarkan berat	Deteksi cepat untuk keberadaan objek jarak dekat	Deteksi objek berdasarkan jarak atau ketinggian di <i>tray</i>

### 3.1.6 Usulan Solusi *Device*

Smartphone menjadi alternatif yang menarik untuk sistem monitoring dan controlling robot pengantar makanan karena ukurannya yang ringkas dan portabilitas yang tinggi. Selain itu, harga yang relatif terjangkau membuat HP banyak dipilih ketika anggaran proyek terbatas. Namun, tampilan data pada layar yang kecil kadang membatasi visualisasi *Dashboard* maupun antarmuka kontrol. Kemampuan multitasking dan pemrosesan data *real-time* pun cukup terbatas [39].

Tablet menempati titik tengah antara HP dan laptop dalam hal ukuran dan kapabilitas. Layar yang lebih luas dibandingkan HP menjadikannya ideal untuk menampilkan informasi serta memudahkan navigasi antarmuka kontrol. Selain itu, bobot yang relatif ringan serta daya tahan baterai yang cukup lama membuat tablet praktis digunakan, bahkan ketika robot perlu beroperasi dalam waktu panjang. Tablet sering kali menjadi pilihan terbaik untuk menyeimbangkan portabilitas, kapasitas pemrosesan, dan kenyamanan visual [40].

Laptop menawarkan performa tertinggi di antara ketiga opsi, terutama ketika harus menangani data dalam jumlah besar atau membutuhkan *Software* pengembangan yang kompleks. Layar yang lebih besar dan kemudahan multitasking membuat laptop unggul. Kendati demikian, ukuran dan bobotnya kerap menjadi kendala untuk dipasang permanen pada robot. Selain itu, konsumsi daya yang tinggi dan harga yang umumnya lebih mahal dapat menjadi pertimbangan khusus dalam konteks efisiensi biaya. Pada tabel 3.5 dipaparkan perbandingan antara ketiga usulan solusi sensor pendekripsi makanan dengan kriteria yang telah

ditetukan. Pada tabel 3.6 dipaparkan perbandingan antara ketiga usulan solusi *Device* dengan kriteria yang telah ditentukan [41].

**Tabel 3. 6 Perbandingan *Device***

Kriteria	Smartphone	Tablet	Laptop
Portabilitas	Sangat tinggi, mudah dibawa dan dipasang	Portabel, tetapi sedikit lebih besar	Rendah, berat dan besar untuk dipasang
Layar	Kecil, terbatas untuk visualisasi kompleks	Sedang, ideal untuk <i>Dashboard</i> dan kontrol	Besar, cocok untuk multitasking
Kemampuan Multitasking	Terbatas	Cukup baik	Sangat baik
Daya Tahan Baterai	Tinggi	Cukup tinggi	Rendah, bergantung pada aktivitas
Kinerja	Terbatas, cocok untuk tugas sederhana	Cukup baik, mendukung aplikasi lebih kompleks	Sangat tinggi, cocok untuk tugas berat
Harga	Relatif terjangkau	Lebih mahal dibandingkan smartphone	Mahal dibandingkan smartphone dan tablet
Cocok Untuk	Proyek dengan anggaran rendah, monitoring sederhana	Aplikasi monitoring kompleks dengan visualisasi kaya	Pengembangan perangkat lunak dan analisis data

### 3.1.7 Usulan Solusi *Framework Frontend Web*

Angular adalah *Framework* berbasis TypeScript yang dikembangkan oleh Google, menawarkan solusi all-in-one seperti routing, manajemen *state*, dan pengujian tanpa memerlukan *Library* tambahan. Dengan strong typing melalui TypeScript dan fitur dependency injection, Angular mendukung pengembangan aplikasi berskala besar dengan manajemen kode yang lebih baik. Didukung pembaruan rutin dari Google, *Framework* ini memiliki roadmap yang jelas. Meski begitu, Angular memiliki curva pembelajaran yang

curam, cenderung berat untuk proyek kecil, dan performa DOM-nya lebih lambat dibandingkan React. *Framework* ini paling cocok untuk proyek besar dengan kebutuhan kompleks dan tim yang sudah terbiasa dengan TypeScript [42] [43].

React.js adalah *Library Frontend* open-source yang dikembangkan oleh Facebook. Kelebihannya meliputi pendekatan berbasis komponen yang membuat kode lebih terstruktur, mudah diatur, dan memungkinkan penggunaan berbagai *Library* pihak ketiga untuk memperluas fungsi. Di sisi lain, ketika aplikasi semakin besar, React memerlukan alat tambahan seperti Redux atau Context API untuk manajemen *state*. Oleh karena itu, React cocok digunakan pada proyek yang membutuhkan kustomisasi tinggi serta aplikasi dengan banyak komponen dinamis seperti *Dashboard* monitoring [42], [43].

Vue.js adalah *Framework* progresif yang ringan, cepat, dan mudah dipelajari dengan dokumentasi yang sangat baik dan fitur bawaan seperti Vue Router dan Vuex. Reactive data binding memungkinkan integrasi otomatis data dengan DOM, sementara komunitas kuat di Asia, termasuk Indonesia, semakin mendukung penggunaannya. Namun, Vue memiliki ekosistem lebih kecil dibandingkan React atau Angular meski komunitas open-source-nya aktif. *Framework* ini cocok untuk proyek kecil hingga menengah yang membutuhkan pengembangan cepat, serta ideal untuk pemula yang ingin belajar *Framework Frontend*. Pada tabel 3.7 dipaparkan perbandingan antara ketiga usulan solusi *Framework Frontend* web dengan kriteria yang telah ditentukan [42], [43].

**Tabel 3.7 Perbandingan *Framework Frontend* Web**

Kriteria	Angular	React.js	Vue.js
Penggunaan Utama	Pengembangan <i>Frontend</i> dengan fitur lengkap (all-in-one)	Pengembangan <i>Frontend</i> berbasis komponen	<i>Framework</i> ringan untuk aplikasi <i>Frontend</i>
Keunggulan	Strong typing dengan TypeScript dan dependency injection	Berbasis komponen, mendukung banyak <i>Library</i>	Ringan, memiliki <i>tools</i> bawaan seperti Vue Router dan Vuex

Kriteria	Angular	React.js	Vue.js
Kekurangan	Kurva pembelajaran curam, performa lebih lambat dan lebih berat untuk proyek kecil	Membutuhkan alat tambahan untuk aplikasi besar	Ekosistem lebih kecil dibandingkan React atau Angular
Cocok Untuk	Proyek besar dengan kebutuhan kompleks dan tim yang terbiasa dengan TypeScript	Proyek dengan kustomisasi tinggi, aplikasi dinamis	Proyek kecil hingga menengah dengan pengembangan cepat

### 3.1.8 Usulan Solusi Monitoring Tegangan

*Voltage sensor* dengan VCC di bawah 25V umumnya merujuk pada modul sensor tegangan berbasis pembagi tegangan (*Voltage divider*), seperti modul berbasis resistor  $30\text{k}\Omega$  dan  $7.5\text{k}\Omega$  yang digunakan bersama mikrokontroler seperti Arduino atau ESP32. Sensor ini biasanya digunakan untuk mengukur tegangan DC hingga maksimum sekitar 25V tergantung pada rasio resistor yang digunakan dan batas maksimum *input ADC* mikrokontroler (misalnya 3.3V). Sensor ini bersifat pasif, tidak memiliki penguatan internal, sehingga akurasinya sangat tergantung pada kualitas resistor dan kalibrasi perangkat lunak.

PZEM-017 adalah sensor yang dirancang untuk memantau parameter *listrik* pada sistem DC, terutama tegangan, arus, dan daya. Untuk pengukuran tegangan, PZEM-017 mampu mengukur tegangan DC hingga 300V tergantung varian, dan memberikan hasil yang relatif akurat karena dilengkapi rangkaian pengkondisi sinyal dan pemrosesan internal. Sensor ini biasanya digunakan untuk sistem tenaga surya, baterai kendaraan *listrik*, atau aplikasi industri yang membutuhkan pemantauan tegangan tinggi secara *real-time* melalui komunikasi serial seperti RS485. Keunggulan utamanya adalah kemampuannya dalam mengukur tegangan tinggi secara langsung tanpa perlu membuat rangkaian pembagi tegangan manual, serta kemudahan integrasinya ke sistem digital.

INA219 merupakan sensor tegangan dan arus digital berbasis I2C yang sangat populer dalam aplikasi monitoring energi berbasis mikrokontroler. Untuk pengukuran tegangan, INA219 mampu mengukur tegangan bus (Vbus) hingga 26V dengan resolusi 12-bit dan akurasi tinggi karena menggunakan penguat diferensial internal dan ADC terintegrasi. Sensor ini juga

mampu mengukur tegangan pada sisi *shunt* (tegangan jatuh pada resistor *shunt*), sehingga bisa digunakan untuk menghitung daya. Keunggulan INA219 terletak pada integrasi fungsi pengukuran tegangan, arus, dan daya dalam satu chip kecil, yang memudahkan desain sistem monitoring yang presisi dan hemat ruang.

Kriteria	<i>Voltage sensor</i> VCC <25V	PZEM-017	INA219
Penggunaan Utama	Pengukuran tegangan DC rendah hingga 25V	Monitoring tegangan DC tinggi dan arus pada sistem besar	Pengukuran tegangan, arus, dan daya pada sistem <i>low-mid</i> V
Keunggulan	Sederhana, murah, mudah digunakan dan dikalibrasi	Akurat untuk tegangan tinggi, cocok untuk RS485	Akurat, mendukung I2C, dapat ukur tegangan & arus sekaligus
Kekurangan	Perlu kalibrasi manual dan pembagi tegangan eksternal	Mahal, komunikasi kompleks, kurang cocok untuk proyek kecil	Rentang tegangan terbatas (maks. 26V), harga relatif mahal
Cocok Untuk	Proyek kecil hingga menengah, pembelajaran, <i>prototyping</i>	Sistem tenaga surya, baterai besar, pemantauan industri	Sistem monitoring presisi rendah-menengah, hemat ruang

### 3.2 Analisis dan Pemilihan Solusi

Pemilihan solusi dilakukan berdasarkan parameter yang mencakup kebutuhan fungsional dan non-fungsional sistem, efisiensi pengembangan, kompatibilitas dengan teknologi yang digunakan, kemudahan integrasi, serta dukungan komunitas atau dokumentasi. Parameter ini dirancang untuk memastikan bahwa solusi yang dipilih mampu memenuhi kebutuhan pengguna, mendukung operasional sistem, dan mempermudah pengelolaan serta pengembangan lebih lanjut.

### **3.2.1 Parameter Pemilihan Solusi**

Solusi dipilih berdasarkan kemampuannya memenuhi kebutuhan fungsional dan non-fungsional. Rincian pemilihan solusi untuk tiap komponen akan dijelaskan dalam bagian ini.

#### **A. Matrix Scoring Mekanisme Pemilihan Solusi**

Di bawah ini adalah pembuatan *scoring matrix* untuk solusi monitoring dan controlling Tel-U Interactive Food Assistant yang terperinci dengan parameter-parameter penilaian untuk setiap kategori. Penilaian dilakukan dengan skala 1–5, di mana nilai ini kemudian dikalikan dengan bobot masing-masing parameter, dan hasil akhirnya dirangkum dalam kolom “Total Score”. Berikut rincian dari setiap parameter-parameternya.

##### **1. Performance**

Parameter ini mengukur kecepatan, stabilitas, dan efisiensi solusi dalam menangani berbagai skenario, dari yang sederhana hingga kompleks:

- 5 (Sangat Baik): Menunjukkan performa sangat cepat, stabil, dan efisien tanpa kendala berarti.
- 4 (Baik): Memberikan performa stabil dan efisien pada sebagian besar skenario, meskipun terdapat sedikit penurunan pada beban tinggi.
- 3 (Cukup Baik): Memadai untuk tugas ringan hingga menengah, tetapi mulai menunjukkan kendala pada skenario yang lebih kompleks.
- 2 (Buruk): Lambat dan kurang *responsif* pada beban yang tinggi atau skenario kompleks.
- 1 (Sangat Buruk): Tidak mampu menangani tugas dasar dengan performa memadai, sering mengalami kegagalan.

##### **2. Ease of Use**

Mengacu pada tingkat kemudahan pengguna dalam mengoperasikan solusi tanpa memerlukan pelatihan khusus:

- 5 (Sangat Mudah): Intuitif dan langsung dapat digunakan tanpa dokumentasi tambahan.
- 4 (Mudah): Mudah digunakan, dengan kebutuhan pembelajaran atau panduan tambahan yang minimal.
- 3 (Cukup Mudah): Memerlukan sedikit waktu belajar tambahan, tetapi masih dapat dioperasikan dengan panduan.

- 2 (Sulit): Memerlukan pelatihan atau keahlian teknis tertentu untuk dapat digunakan.
- 1 (Sangat Sulit): Kompleks dan membutuhkan usaha besar untuk dipahami, bahkan dengan panduan.

### **3. *Community Support***

Dukungan komunitas mengukur sejauh mana ekosistem pengguna dan pengembang membantu menyelesaikan masalah atau mendukung pengembangan:

- 5 (Sangat Baik): Memiliki komunitas besar, aktif, dan menyediakan sumber daya yang luas, seperti forum, tutorial, dan dokumentasi.
- 4 (Baik): Komunitas cukup besar dan aktif, dengan dukungan memadai untuk sebagian besar masalah.
- 3 (Cukup Baik): Komunitas terbatas tetapi memberikan dukungan untuk masalah umum.
- 2 (Buruk): Komunitas kecil dan jarang aktif, dengan sumber daya yang sulit diakses.
- 1 (Sangat Buruk): Hampir tidak ada komunitas pendukung, membuat dukungan teknis sulit ditemukan.

### **4. *Integration***

Mengukur kemampuan solusi untuk terintegrasi dengan teknologi atau platform lain:

- 5 (Sangat Baik): Sepenuhnya kompatibel dengan berbagai platform tanpa memerlukan penyesuaian tambahan.
- 4 (Baik): Kompatibel dengan sebagian besar platform dengan sedikit usaha tambahan.
- 3 (Cukup Baik): Kompatibel, tetapi membutuhkan konfigurasi tambahan untuk integrasi penuh.
- 2 (Buruk): Sulit diintegrasikan dengan teknologi lain, memerlukan usaha signifikan.
- 1 (Sangat Buruk): Hampir tidak mendukung integrasi dengan platform lain.

### **5. *Cost***

Parameter ini mengukur keterjangkauan biaya solusi dibandingkan dengan fungsionalitas yang ditawarkan:

- 5 (Sangat Terjangkau): Memberikan nilai luar biasa dengan biaya rendah atau tanpa biaya tambahan.

- 4 (Terjangkau): Biaya sebanding dengan fungsionalitas yang disediakan.
- 3 (Cukup Terjangkau): Ekonomis, tetapi terdapat keterbatasan dalam fitur atau performa.
- 2 (Mahal): Membutuhkan investasi besar dengan manfaat yang kurang optimal.
- 1 (Sangat Mahal): Biaya sangat tinggi tanpa manfaat yang memadai.

Setelah menentukan parameter-parameternya kemudian pada bagian ini dilakukan perhitungan *Matrix Scoring* untuk setiap usulan solusi yang telah disebutkan sebelumnya.

## B. *Mobile Development Framework*

Tabel 3.8 menunjukkan hasil perhitungan *Matrix Scoring* untuk pemilihan solusi *mobile development Framework*, dengan pilihan terbaik yaitu Android Studio.

**Tabel 3.8 Matrix Scoring Mobile Development Framework**

Parameter	Bobot	Kotlin		Flutter		React Native	
		Nilai	Nilai x Bobot	Nilai	Nilai x Bobot	Nilai	Nilai x Bobot
<i>Performance</i>	20%	5	1	5	1	3	0.6
<i>Ease of Use</i>	20%	4	0.8	3	0.6	5	1
<i>Community Support</i>	20%	5	1	5	1	4	0.8
<i>Integration</i>	20%	5	1	5	1	4	0.8
<i>Cost</i>	20%	5	1	5	1	5	1
<b>Total Score (sum)</b>	100%	24	4.8	23	4.6	21	4.2

Kotlin dipilih sebagai solusi karena merupakan pengembangan aplikasi Android yang memberikan integrasi mendalam dengan Android SDK. Dengan fitur-fitur eksklusif seperti emulator bawaan, *debugging* yang kuat, dan dukungan terhadap bahasa pemrograman Kotlin dan Java, Android Studio menawarkan kemampuan terbaik untuk membangun aplikasi native

dengan performa optimal. IDE ini juga ideal untuk aplikasi yang membutuhkan akses mendalam ke sistem Android dan optimasi tinggi, seperti pengendalian robot pengantar makanan.

### C. Database

Tabel 3.9 menunjukkan hasil perhitungan *Matrix Scoring* untuk pemilihan solusi database, dengan pilihan terbaik yaitu MySQL.

**Tabel 3.9 Matrix Scoring Database**

<b>Parameter</b>	<b>Bobot</b>	<b>MySQL</b>		<b>Firebase</b>		<b>MongoDB</b>	
		<b>Nilai</b>	<b>Nilai x Bobot</b>	<b>Nilai</b>	<b>Nilai x Bobot</b>	<b>Nilai</b>	<b>Nilai x Bobot</b>
<i>Performance</i>	20%	4	1	5	1	3	0.6
<i>Ease of Use</i>	20%	5	0.8	5	0.6	3	0.6
<i>Community Support</i>	20%	5	1	5	1	4	0.8
<i>Integration</i>	20%	4	1	4	1	3	0.6
<i>Cost</i>	20%	5	1	3	1	4	0.6
<b>Total Score (sum)</b>	100%	23	4.6	22	4.4	17	3.1

MySQL dipilih karena kemampuannya dalam menangani data terstruktur dengan efisien melalui *query SQL* yang kuat. Database ini sangat cocok untuk menyimpan informasi seperti riwayat pengiriman, status robot, dan data pengguna yang memerlukan operasi CRUD kompleks serta relasi antar tabel. Selain itu, MySQL memiliki stabilitas yang tinggi dan telah

terbukti andal dalam aplikasi yang membutuhkan pengolahan data skala kecil hingga menengah.

#### D. Mikrokontroller

Tabel 3.10 menunjukkan hasil perhitungan *Matrix Scoring* untuk pemilihan solusi mikrokontroller, dengan pilihan terbaik yaitu ESP32.

**Tabel 3. 10 *Matrix Scoring* Mikrokontroller**

Parameter	Bobot	ESP32		ESP8266		Arduino	
		Nilai	Nilai x Bobot	Nilai	Nilai x Bobot	Nilai	Nilai x Bobot
<b>Performance</b>	20%	5	1	4	0.8	5	1
<b>Ease of Use</b>	20%	5	1	5	1	5	1
<b>Community Support</b>	20%	5	1	5	1	5	1
<b>Integration</b>	20%	5	1	4	0.8	3	0.6
<b>Cost</b>	20%	4	0.8	4	0.8	3	0.6
<b>Total Score (sum)</b>	100%	24	4.6	22	4.4	21	4.2

ESP32 seperti pada gambar 3.1 menjadi pilihan utama karena memiliki kemampuan Wi-Fi dan Bluetooth yang terintegrasi, prosesor dual-core dengan kecepatan tinggi, serta mendukung berbagai antarmuka seperti GPIO, SPI, dan I2C. Hal ini membuat ESP32 sangat fleksibel untuk mengelola berbagai sensor dan aktuator dalam aplikasi IoT seperti robot pengantar makanan. Selain itu, kemampuan multitasking dan kompatibilitasnya dengan berbagai *Library* menjadikan ESP32 sebagai solusi efisien untuk pengembangan sistem pintar.



**Gambar 3. 1 ESP32S3**

#### E. Backend Framework

Tabel 3.11 menunjukkan hasil perhitungan *Matrix Scoring* untuk pemilihan solusi *backend Framework*, dengan pilihan terbaik yaitu Node.js.

**Tabel 3. 11 Matrix Scoring Backend Framework**

Parameter	Bobot	Node JS		Django		Spring Boot	
		Nilai	Nilai x Bobot	Nilai	Nilai x Bobot	Nilai	Nilai x Bobot
<b>Performance</b>	20%	5	1	5	1	4	0.8
<b>Ease of Use</b>	20%	4	0.8	4	0.8	3	0.6
<b>Community Support</b>	20%	5	1	5	1	5	1
<b>Integration</b>	20%	5	1	4	0.8	3	0.6
<b>Cost</b>	20%	5	1	5	1	5	1
<b>Total Score (sum)</b>	100%	24	4.8	23	4.6	20	4

Node.js dipilih karena kemampuannya dalam menangani komunikasi *real-time* dengan efisiensi tinggi melalui model *non-blocking*. Hal ini sangat penting untuk mengontrol robot

pengantar makanan secara langsung dan memperbarui status robot secara instan. Dukungan terhadap protokol seperti WebSocket dan MQTT membuat Node.js ideal untuk membangun *backend* yang mendukung komunikasi interaktif dan sinkronisasi data secara *real-time*.

#### **F. Device**

Tabel 3.12 menunjukkan hasil perhitungan *Matrix Scoring* untuk pemilihan solusi *Device*, dengan pilihan terbaik yaitu tablet.

**Tabel 3. 12 Matrix Scoring Device**

<b>Parameter</b>	<b>Bobot</b>	<b>Tablet</b>		<b>Smartphone</b>		<b>Laptop</b>	
		<b>Nilai</b>	<b>Nilai x Bobot</b>	<b>Nilai</b>	<b>Nilai x Bobot</b>	<b>Nilai</b>	<b>Nilai x Bobot</b>
<b>Performance</b>	20%	5	1	4	0.8	5	1
<b>Ease of Use</b>	20%	5	1	5	1	5	1
<b>Community Support</b>	20%	5	1	5	1	5	1
<b>Integration</b>	20%	5	1	4	0.8	3	0.6
<b>Cost</b>	20%	4	0.8	5	1	3	0.6
<b>Total Score (sum)</b>	100%	24	4.8	23	4.6	21	4.1

Tablet seperti yang ditampilkan pada gambar 3.2 dipilih sebagai perangkat untuk monitoring dan controlling karena layarnya yang lebih besar dibandingkan smartphone, namun tetap memiliki portabilitas tinggi dibandingkan laptop. Dengan kapasitas baterai yang memadai dan performa cukup baik untuk menjalankan aplikasi monitoring, tablet menjadi pilihan terbaik untuk menampilkan *Dashboard* dan navigasi antarmuka kontrol secara visual.



**Gambar 3. 2 Tablet**

#### **G. Sensor Pendekripsi Makanan**

Tabel 3.13 menunjukkan hasil perhitungan *Matrix Scoring* untuk pemilihan solusi sensor pendekripsi makanan, dengan pilihan terbaik yaitu IR sensor.

**Tabel 3. 13 *Matrix Scoring* Sensor Pendekripsi Makanan**

<b>Parameter</b>	<b>Bobot</b>	<b>IR</b>		<b>Ultrasonik</b>		<b>Load</b>	
		<b>Nilai</b>	<b>Nilai x Bobot</b>	<b>Nilai</b>	<b>Nilai x Bobot</b>	<b>Nilai</b>	<b>Nilai x Bobot</b>
<b>Performance</b>	20%	5	1	5	1	5	1
<b>Ease of Use</b>	20%	5	1	5	1	5	1
<b>Community Support</b>	20%	5	1	5	1	5	1
<b>Integration</b>	20%	5	1	4	0.8	4	0.8
<b>Cost</b>	20%	5	1	4	0.8	4	0.8
<b>Total Score (sum)</b>	100%	25	5	23	4.6	23	4.8

IR Sensor seperti pada gambar 3.3 dipilih karena kemampuannya mendekripsi keberadaan objek secara cepat tanpa kontak fisik. Teknologi ini cocok untuk mendekripsi makanan pada *tray* robot pengantar, dengan keunggulan harga yang terjangkau dan kemudahan integrasi

dengan mikrokontroler seperti ESP32. Meski sensitif terhadap cahaya sekitar, IR Sensor tetap menjadi solusi yang efisien untuk kebutuhan deteksi jarak dekat.



**Gambar 3. 3 IR Sensor**

#### H. *Web Development Framework*

Tabel 3.14 menunjukkan hasil perhitungan *Matrix Scoring* untuk pemilihan solusi web development *Framework*, dengan pilihan terbaik yaitu Vue.js.

**Tabel 3. 14 Matrix Scoring Web Development Framework**

Parameter	<b>Bobot</b>	Vue JS		React JS		Angular	
		Nilai	Nilai x <b>Bobot</b>	Nilai	Nilai x <b>Bobot</b>	Nilai	Nilai x <b>Bobot</b>
<b>Performance</b>	20%	5	1	4	0.8	4	0.8
<b>Ease of Use</b>	20%	5	1	4	0.8	4	0.8
<b>Community Support</b>	20%	4	0.8	4	0.8	4	0.8
<b>Integration</b>	20%	4	0.8	5	1	4	0.8
<b>Cost</b>	20%	5	1	5	1	5	1
<b>Total Score (sum)</b>	100%	23	4.8	22	4.6	21	4.2

Vue.js merupakan pilihan yang paling sesuai, *Framework* ini ringan dan cepat, sehingga ideal untuk aplikasi monitoring. Selain itu, kesederhanaan Vue.js memudahkan proses

pengembangan, terutama jika proyek memiliki batasan waktu. Vue juga dilengkapi dengan *tools* bawaan seperti Vue Router untuk routing dan Vuex untuk manajemen *state* yang cukup untuk menangani kebutuhan monitoring tanpa harus menggunakan *Library* tambahan.

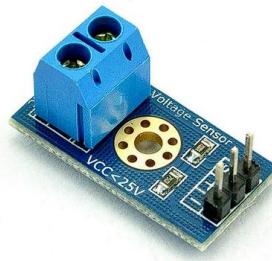
## I. Sensor Tegangan

Tabel 3.15 menunjukkan hasil perhitungan *Matrix Scoring* untuk pemilihan solusi Sensor Tegangan, dengan pilihan terbaik yaitu *Voltage sensor VCC <25V*.

**Tabel 3.15 Matrix Scoring Sensor Tegangan**

<b>Parameter</b>	<b>Bobot</b>	<i>Voltage sensor</i> VCC <25V		<b>INA219</b>		<b>PZEM</b>	
		<b>Nilai</b>	<b>Nilai x Bobot</b>	<b>Nilai</b>	<b>Nilai x Bobot</b>	<b>Nilai</b>	<b>Nilai x Bobot</b>
<b><i>Performance</i></b>	20%	5	1	5	1	5	1
<b><i>Ease of Use</i></b>	20%	5	1	4	0.8	4	0.8
<b><i>Community Support</i></b>	20%	4	0.8	4	0.8	4	0.8
<b><i>Integration</i></b>	20%	4	0.8	4	0.8	5	1
<b><i>Cost</i></b>	20%	5	1	5	1	3	0.6
<b><i>Total Score (sum)</i></b>	100%	23	4.8	22	4.6	21	4.2

*Voltage sensor VCC <25V* merupakan pilihan yang paling sesuai, sensor ini ringan dan cukup stabil, sehingga ideal untuk pengukuran tegangan pada baterai secara real-time. Selain itu, kemudahan pengimplementasian sensor ini menjadi salah satu keunggulannya, terutama jika proyek memiliki batasan waktu. Sensor ini juga dilengkapi dengan *Voltage divider* yang bisa membagi tegangan yang masuk kedalam sensor ini yang membuat sensor ini hanya dapat mengeluarkan output atau keluaran sebesar 5V.



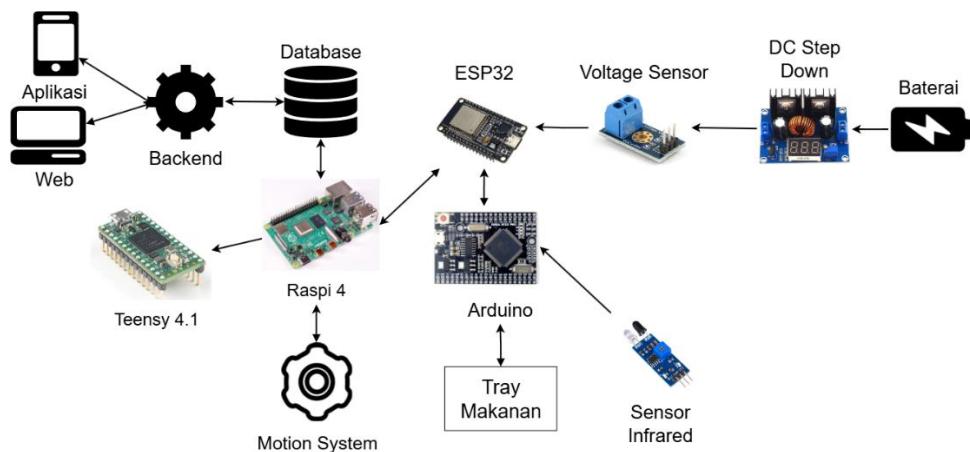
**Gambar 3. 4 *Voltage sensor***

### 3.3 Desain Solusi Terpilih

Desain solusi yang dipilih dirancang untuk memenuhi kebutuhan fungsional dan non-fungsional sistem berdasarkan parameter yang telah ditetapkan sebelumnya. Solusi ini mencakup arsitektur sistem, *Flowchart* pada perangkat keras, *Flowchart* pada perangkat lunak dan integrasi antar sistem dan komponen untuk memastikan kelancaran operasional serta efisiensi dalam pengembangan dan pengelolaan sistem. Pendekatan desain ini difokuskan untuk memberikan solusi yang optimal dan berorientasi pada kebutuhan pengguna.

#### 3.3.1 Arsitektur Rancangan Umum

Gambar 3.5 menampilkan arsitektur rancangan umum pada Robot Tel-U Interactive Food Assitant.



**Gambar 3. 5 Arsitektur Rancangan Umum**

Berikut adalah penjelasan singkat mengenai diagram sistem pada gambar 3.5.

### **A. Pusat Kendali (Teensy 4.1)**

Teensy 4.1 berperan sebagai otak utama yang mengkoordinasikan data dan instruksi antar berbagai modul. Kinerja prosesor yang cepat di Teensy 4.1 membuatnya cocok untuk menangani banyak sinyal sensor dan pemrosesan *real time*.

### **B. Komunikasi dengan Sensor dan Modul Lain**

Bagian ini menjelaskan peran masing-masing modul dalam mengirim dan menerima data sensor, serta bagaimana perangkat saling berkomunikasi untuk mendukung kinerja sistem secara *real-time*.

#### **1. ESP32**

- Membaca sensor tegangan untuk memantau status baterai. Nilai tegangan yang diperoleh dikonversi ke dalam bentuk persentase dan dikirimkan ke *backend* untuk disimpan dalam database.
- Mengirimkan data ke *backend* melalui Raspi, untuk ditampilkan pada aplikasi *mobile* dan web monitoring.

#### **2. Arduino**

Arduino terhubung langsung dengan sensor infrared dan *tray* makanan. Mikrokontroler ini bertugas membaca data dari sensor untuk mengetahui kondisi *tray* (kosong atau terisi), kemudian mengirimkan data tersebut ke Raspberry Pi untuk diproses lebih lanjut. Modul IoT.

#### **3. Raspberry Pi**

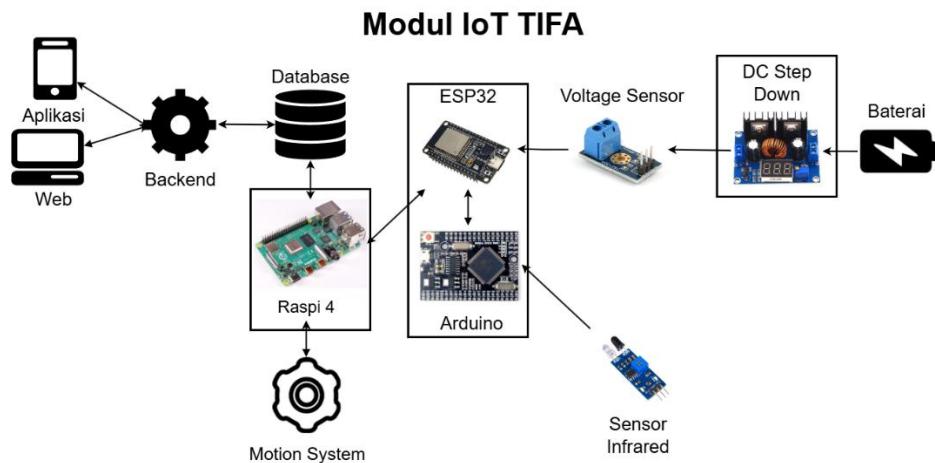
Raspberry Pi 4 berperan sebagai pengelola komunikasi antar mikrokontroler dan menjadi jembatan antara perangkat keras dan *backend*. Dengan kapabilitas komputasi yang lebih tinggi dibandingkan mikrokontroler biasa, Raspberry Pi mampu menangani proses pengumpulan, pengolahan awal, dan pengiriman data secara efisien.

Dalam sistem ini, Raspberry Pi memiliki beberapa fungsi utama. Pertama, Raspberry Pi mengelola data yang dikirimkan oleh Arduino dan ESP32, termasuk status *tray* makanan dan tingkat daya baterai. Kedua, Raspberry Pi menghubungkan sistem mikrokontroler ke *backend* dengan meneruskan data yang diterima ke database melalui koneksi internet. Ketiga, Raspberry Pi dapat mengirimkan perintah ke Teensy 4.1 sebagai pengendali gerakan, apabila diperlukan, berdasarkan hasil analisis data sensor.

Selain itu, Raspberry Pi juga berfungsi sebagai pusat integrasi untuk semua sinyal *input* dan *output* dari sensor maupun aktuator dalam sistem. Peran ini sangat penting untuk memastikan sinkronisasi antar perangkat berjalan lancar, sehingga keseluruhan sistem dapat

bekerja secara *real-time* dan otomatis dalam proses pengantaran makanan. Dengan arsitektur ini, sistem IoT TIFA mampu bekerja secara *real-time* dan efisien dalam mengantarkan makanan secara otomatis dan cerdas [44].

### 3.3.2 Arsitektur Modul IoT TIFA



**Gambar 3. 6 Arsitektur Rancangan Modul IoT**

Gambar 3.6 di atas merupakan ilustrasi arsitektur sistem dari Modul IoT TIFA (Tel-U Interactive Food Assistant), yang menunjukkan hubungan antar komponen utama sistem pemantauan dan pengendalian perangkat secara otomatis. Komponen yang ditampilkan di dalam kotak merupakan bagian dari ranah kerja divisi lain yang mendukung integrasi sistem secara keseluruhan, yang dirancang untuk memantau kondisi baterai serta mengendalikan perangkat secara otomatis dan terintegrasi. Sistem ini mengadopsi konsep *Internet of Things* (IoT), yang menghubungkan berbagai komponen perangkat keras dan perangkat lunak melalui jaringan (WiFi) agar dapat saling berkomunikasi dan bertukar data secara *real-time*.

#### A. Baterai

Baterai merupakan sumber daya utama dari sistem TIFA. Tegangan baterai ini dipantau secara berkala untuk memastikan bahwa perangkat tetap dapat beroperasi dengan aman dan efisien. Pemantauan ini penting untuk mencegah kondisi *over-discharge* yang dapat merusak sel baterai atau menyebabkan sistem mati mendadak.

#### B. Voltage sensor

Tegangan dari baterai terlebih dahulu diukur menggunakan modul sensor tegangan (*Voltage sensor*). Sensor ini berfungsi sebagai pembagi tegangan (*Voltage divider*) yang mengamankan sinyal agar dapat diterima oleh pin analog ESP32. Nilai tegangan yang terbaca akan dikalibrasi untuk mendapatkan hasil pengukuran yang akurat.

## C. ESP32

ESP32 merupakan mikrokontroler utama yang menangani komunikasi antara sensor dan server. Fungsi utamanya adalah:

1. Membaca data dari *Voltage sensor*.
2. Menghitung nilai tegangan aktual dan persentase kapasitas baterai.
3. Mengirim data tegangan ke server (*backend*) melalui POST.

## D. Sensor Infrared

Sensor infrared digunakan untuk mendeteksi keberadaan makanan atau minuman di dalam *tray*. Sensor ini berfungsi sebagai komponen *input* tambahan yang juga dapat memicu aksi tertentu pada sistem berdasarkan logika yang diatur di *backend*.

## E. Backend Server

*Backend* merupakan bagian penting dari sistem yang berfungsi sebagai pengelola data dan logika utama. Fungsinya meliputi:

1. Menerima data tegangan dan status sensor dari ESP32 dan Raspberry Pi.
2. Menyimpan data ke dalam database untuk keperluan monitoring dan histori.
3. Menyediakan API *endpoint* yang dapat diakses oleh aplikasi dan web.
4. Menyusun logika sistem seperti validasi status, pengendalian perangkat, dan otorisasi pengguna.

*Backend* ini dibangun menggunakan bahasa server-side seperti PHP dan berkomunikasi dengan database menggunakan *query SQL*.

## F. Database

Database digunakan untuk menyimpan seluruh data yang dibutuhkan sistem, seperti:

1. Data tegangan baterai
2. Status perangkat
3. Riwayat aktivitas dan log pengguna.
4. Perintah yang dikirim dari antarmuka aplikasi/web

Data ini dapat digunakan untuk kebutuhan monitoring, analisis, hingga visualisasi di aplikasi dan website.

## G. Aplikasi Android dan Website

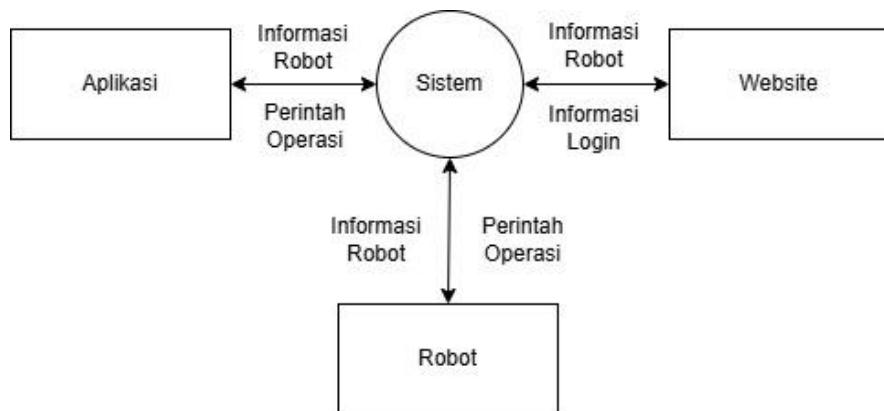
Aplikasi dan website merupakan antarmuka pengguna (*user interface*) yang memungkinkan operator atau pengguna akhir untuk:

1. Melihat status baterai secara *real-time*.
2. Mengirim perintah ke sistem (misalnya membuka atau menutup pintu otomatis).
3. Melihat data histori atau notifikasi dari sistem.
4. Melihat riwayat penggunaan dan status sistem.

Semua interaksi dari aplikasi dan website mengakses data dan layanan dari *backend* yang terhubung.

### 3.3.3 Desain *Flow Diagram* Fungsional

*Data Flow Diagram* (DFD) Level 0 pada gambar 3.7 di bawah ini menggambarkan aliran data utama antara Web Monitoring, Aplikasi, dan Robot, dengan Sistem sebagai pusat pengelolaan data dan komunikasi. Web Monitoring berfungsi sebagai antarmuka bagi pengguna untuk *login* dan menerima informasi terkait Status Robot secara *real-time*. Web Monitoring mengirimkan Informasi *Login* ke Sistem untuk autentikasi, dan Sistem memberikan umpan balik berupa Status *Login* (berhasil/gagal) serta Status Robot setelah *login* berhasil.



**Gambar 3. 7 Desain *Flow Diagram* Fungsional**

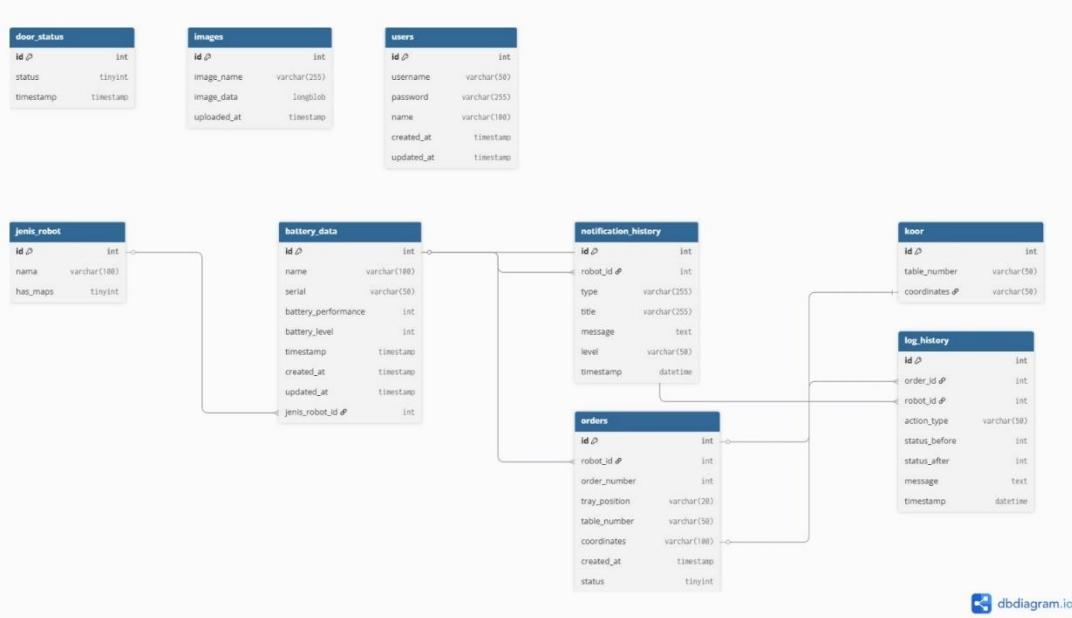
Aplikasi digunakan untuk mengendalikan operasional robot dengan mengirimkan Perintah Operasi ke Sistem. Sebagai tanggapan, Sistem mengirimkan Status Operasi ke Aplikasi untuk memantau aktivitas robot. Robot berperan sebagai entitas fisik yang menerima Perintah Operasi dari Sistem dan memberikan Status Robot sebagai *feedback* atas perintah tersebut.

Sistem bertindak sebagai mediator yang mengelola alur data antara Web Monitoring, Aplikasi, dan Robot. Sistem memproses data *login* dari Web Monitoring, mengirimkan status *login*, serta menyediakan data operasional robot ke semua pihak yang relevan. Aliran data yang

terjadi mencakup komunikasi dua arah antara Sistem dan Aplikasi, serta Sistem dan Robot, sementara Web Monitoring hanya menerima data berupa status *login* dan status robot dari Sistem. Diagram ini mencerminkan hubungan dan aliran data yang jelas tanpa merinci proses internal Sistem.

### 3.3.4 Entity Relationship Diagram Database

Database dbtes terdiri atas tujuh tabel utama yang saling berhubungan seperti yang digambarkan pada gambar 3.8, masing-masing memiliki peran spesifik dalam pengelolaan sistem monitoring dan pengendalian robot berbasis data.



Gambar 3.8 Entity Relationship Diagram Database

Adapun penjelasan masing-masing tabel adalah sebagai berikut:

#### A. Tabel battery\_data

Tabel battery\_data menyimpan informasi penting mengenai status baterai dari setiap robot. Setiap entri memuat identitas robot seperti nama dan nomor seri, serta informasi teknis seperti performa baterai, level baterai, waktu pencatatan, dan waktu pembaruan. Kolom jenis\_robot\_id menghubungkan robot dengan jenisnya yang terdapat di tabel jenis\_robot. Tabel ini menjadi inti dalam sistem karena mewakili identitas dan kondisi setiap unit robot yang digunakan.

## **B. Tabel door\_status**

Tabel ini digunakan untuk mencatat status pintu, seperti apakah pintu dalam kondisi terbuka atau tertutup. Status dicatat dalam bentuk nilai tinyint, dan setiap perubahan dicatat dengan waktu (*timestamp*). Data ini sangat berguna untuk sistem otomatisasi yang membutuhkan kontrol akses.

## **C. Tabel images**

Tabel images menyimpan data gambar, seperti peta area atau hasil tangkapan kamera. Setiap gambar disimpan dengan nama (*image\_name*), data gambar dalam format biner (*image\_data*), dan waktu pengunggahan (*uploaded\_at*). Tabel ini mendukung fungsi visualisasi atau dokumentasi sistem yang menggunakan gambar dalam operasionalnya.

## **D. Tabel users**

Tabel ini menyimpan data pengguna sistem, seperti operator atau administrator. Setiap pengguna memiliki *username*, *password*, dan nama lengkap, serta waktu pembuatan dan pembaruan akun. Sistem ini memungkinkan otentikasi dan manajemen hak akses terhadap fungsi-fungsi dalam sistem berdasarkan identitas pengguna.

## **E. Tabel jenis\_robot**

Tabel jenis\_robot mencatat jenis atau tipe dari setiap robot yang ada di sistem. Setiap entri berisi nama jenis robot dan informasi apakah robot tersebut memiliki kemampuan peta (*has\_maps*). Tabel ini dihubungkan dengan *battery\_data*, memungkinkan sistem mengetahui tipe robot dan fitur yang dimilikinya.

## **F. Tabel notification\_history**

Tabel ini digunakan untuk menyimpan riwayat notifikasi yang dikirim ke pengguna atau ditampilkan di sistem. Informasi yang disimpan meliputi jenis notifikasi, judul, isi pesan, level prioritas, dan waktu pengiriman. Setiap notifikasi terhubung ke robot tertentu melalui kolom *robot\_id*, sehingga sistem dapat melacak histori peringatan atau informasi penting yang berkaitan dengan masing-masing robot.

## **G. Tabel orders**

Tabel orders mencatat setiap pesanan atau tugas yang diberikan kepada robot. Informasi yang tersimpan termasuk robot yang menerima tugas, nomor pesanan, posisi nampan

(*tray\_position*), nomor meja tujuan, koordinat lokasi, waktu dibuatnya order, dan status order tersebut. Tabel ini menjadi pusat dalam pengaturan tugas robot, dan terhubung dengan tabel koor untuk mencocokkan koordinat lokasi dengan titik-titik nyata dalam peta.

## H. Tabel koor

Tabel koor menyimpan informasi koordinat lokasi dalam sistem, termasuk nomor meja dan nilai koordinat yang digunakan dalam peta. Kolom coordinates dibuat unik agar bisa dihubungkan secara one-to-one dengan kolom coordinates di tabel orders. Hal ini memungkinkan sistem mengaitkan pesanan ke titik lokasi yang spesifik, seperti meja tujuan robot pengantar makanan.

## I. Tabel log\_history

Tabel ini berfungsi untuk mencatat semua aktivitas dan perubahan status yang terjadi saat robot menjalankan tugas. Setiap log menyimpan ID pesanan (*order\_id*), ID robot (*robot\_id*), tipe aksi (*action\_type*), status sebelumnya dan sesudahnya, serta pesan atau catatan terkait aktivitas tersebut. Informasi ini sangat penting untuk audit trail dan pemantauan histori operasional robot secara Detail.

### 3.3.5 Flowchart

Untuk menggambarkan alur kerja sistem, *Flowchart* disusun sebagai representasi visual proses utama, membantu memahami hubungan antar komponen dan langkah-langkah sistem. Rincian *Flowchart* aplikasi dan web monitoring akan dijelaskan pada sub bab berikut.

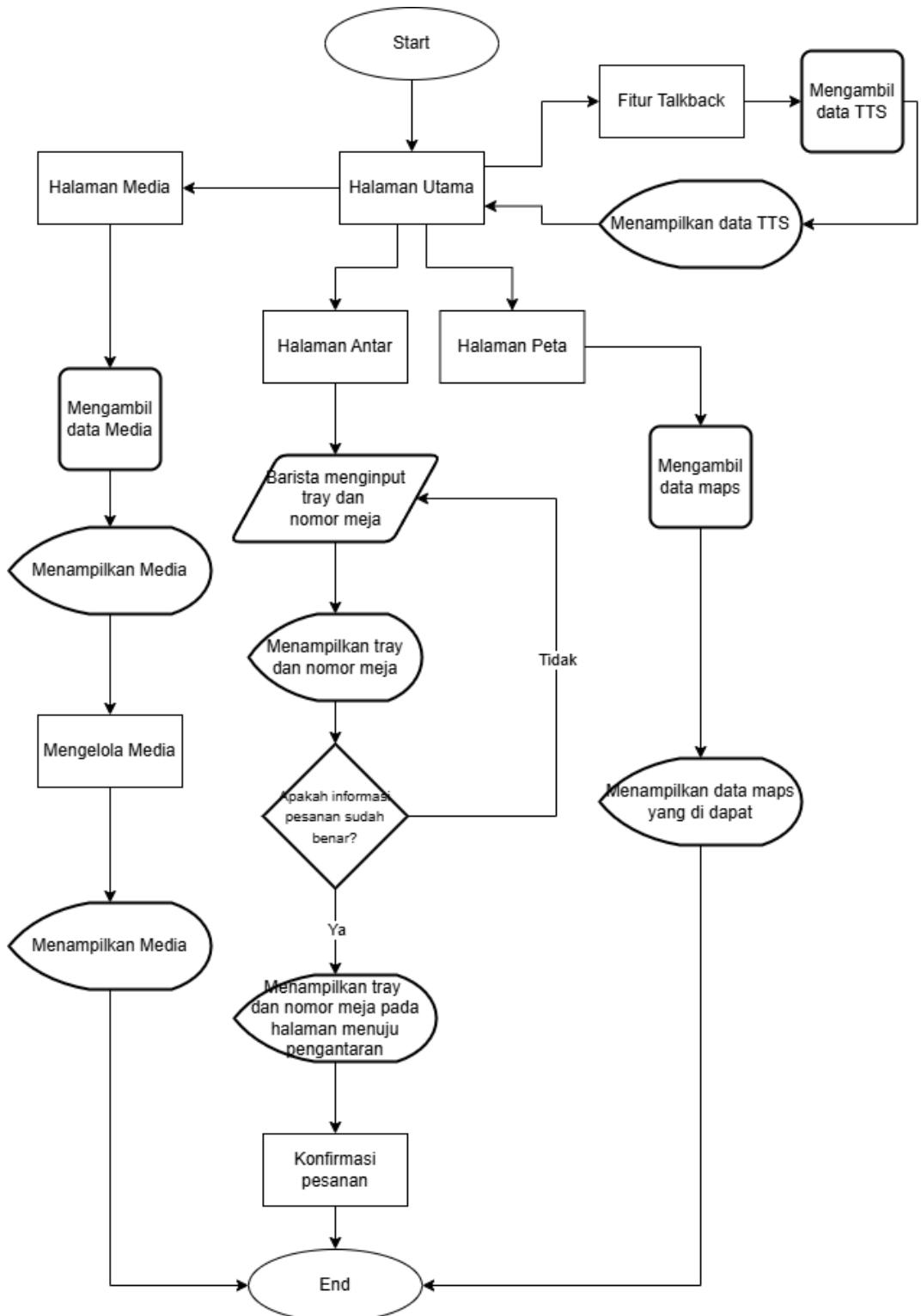
#### A. Flowchart Aplikasi

Gambar 3.9 di bawah ini menggambarkan alur kerja utama dalam aplikasi yang digunakan oleh Barista. Proses dimulai dari start dan memiliki beberapa cabang utama, yaitu mode Antar, fitur Media, fitur Peta, fitur *TalkBack*.

1. Fitur Media: Jika Barista memilih untuk masuk ke halaman media, aplikasi akan mengambil data media yang tersedia, menampilkannya, dan memberikan opsi untuk mengelola media tersebut sebelum ditampilkan kembali.
2. Fitur Antar: Pada mode antar, Barista menginput *tray* dan nomor meja. Aplikasi kemudian menampilkan data tersebut untuk diverifikasi. Jika informasi belum benar, Barista dapat mengulangi proses *input*. Jika sudah benar, informasi *tray* dan nomor meja akan ditampilkan pada halaman menuju pengantaran, dilanjutkan dengan proses konfirmasi pesanan untuk menyelesaikan pengantaran.

3. Fitur Peta: Jika Barista memilih halaman peta, aplikasi akan mengambil data peta dan menampilkannya kepada pengguna. Tidak disebutkan adanya opsi penggantian peta atau konfirmasi lanjutan dalam *Flowchart*, sehingga proses berakhir setelah data peta ditampilkan.
4. Fitur *TalkBack*: Fitur ini mengambil data TTS (Text-to-Speech) dan menampilkannya pada halaman utama, memungkinkan pengguna mendengar informasi dengan bantuan suara.

*Flowchart* ini menunjukkan alur interaksi yang terstruktur dalam aplikasi, memastikan setiap langkah, khususnya yang berkaitan dengan *input* data, diverifikasi terlebih dahulu sebelum melanjutkan ke proses selanjutnya. Dengan demikian, aplikasi menawarkan fleksibilitas dalam penggunaan dan memberikan pengalaman pengguna yang efisien serta mudah dipahami.



**Gambar 3. 9 Flowchart Aplikasi**

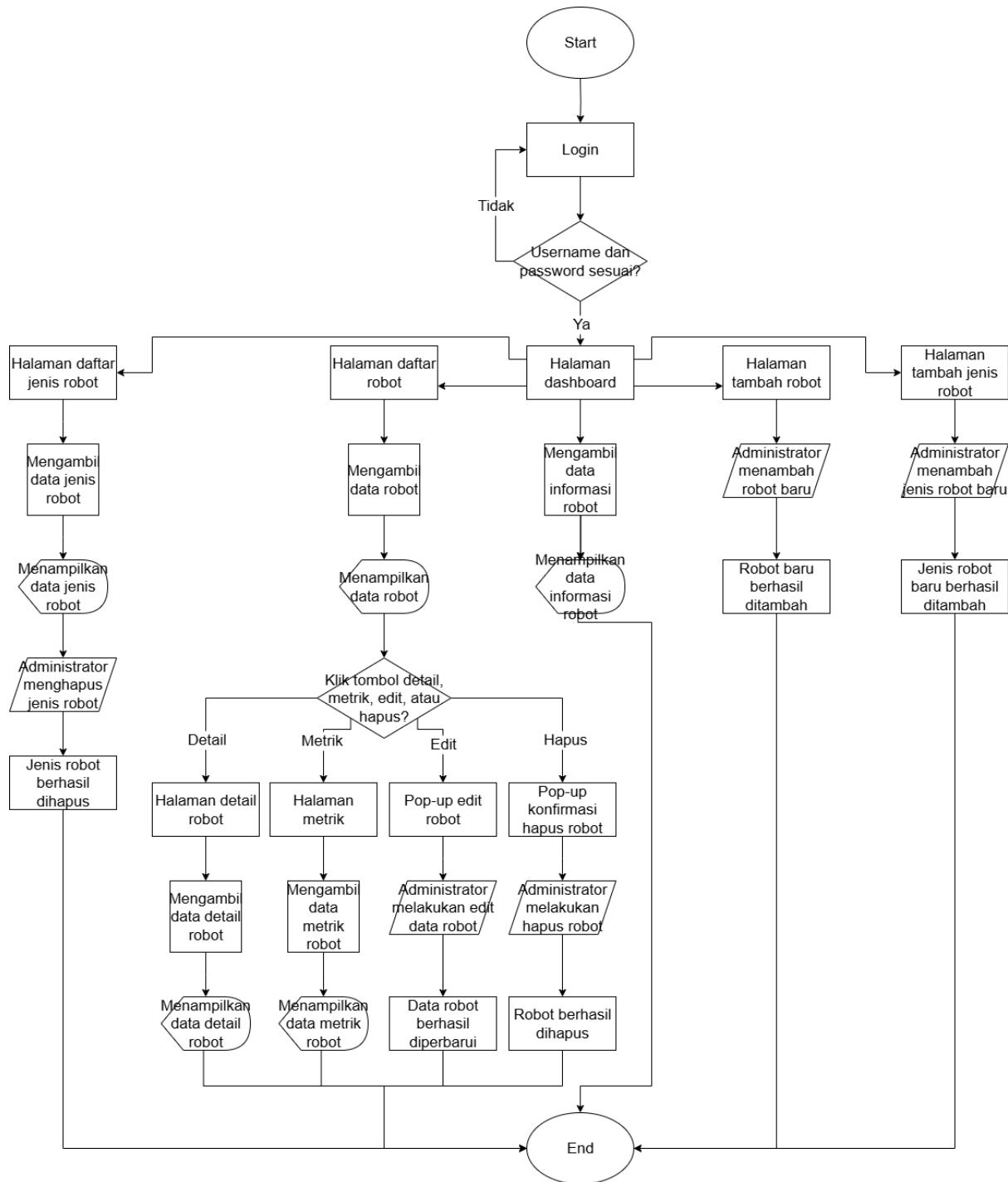
## B. *Flowchart* Web Monitoring

*Flowchart* pada gambar 3.10 menggambarkan alur kerja web monitoring, dimulai dari proses *login* hingga manajemen data robot dan jenis robot. Proses dimulai dari pengguna membuka halaman *login* dan memasukkan *username* serta *password*. Jika informasi *login* tidak sesuai, pengguna akan diarahkan kembali ke halaman *login*. Namun jika sesuai, pengguna akan masuk ke halaman *Dashboard* yang menampilkan informasi umum tentang robot.

Dari *Dashboard*, pengguna memiliki beberapa jalur navigasi. Pertama, pengguna dapat masuk ke halaman daftar robot untuk melihat semua robot yang terdaftar. Pada halaman ini, pengguna dapat memilih untuk melihat Detail robot, metrik, mengedit data robot, atau menghapus robot. Aksi tersebut dilakukan melalui tombol aksi yang menampilkan halaman Detail, halaman metrik, *Pop-up* edit robot, atau *Pop-up* konfirmasi penghapusan.

Selain itu, pengguna juga dapat mengakses halaman tambah robot untuk menambahkan robot baru. Setelah administrator mengisi informasi yang dibutuhkan dan menyimpannya, robot baru berhasil ditambahkan ke sistem. Hal serupa juga berlaku untuk fitur tambah jenis robot, di mana pengguna dapat membuat kategori robot baru.

Terakhir, halaman daftar jenis robot menampilkan semua kategori robot yang sudah dibuat. Dari sini, administrator dapat menghapus jenis robot yang tidak lagi diperlukan. Semua jalur aktivitas dalam sistem ini akhirnya mengarah ke akhir proses, menunjukkan bahwa alur kerja sudah selesai.



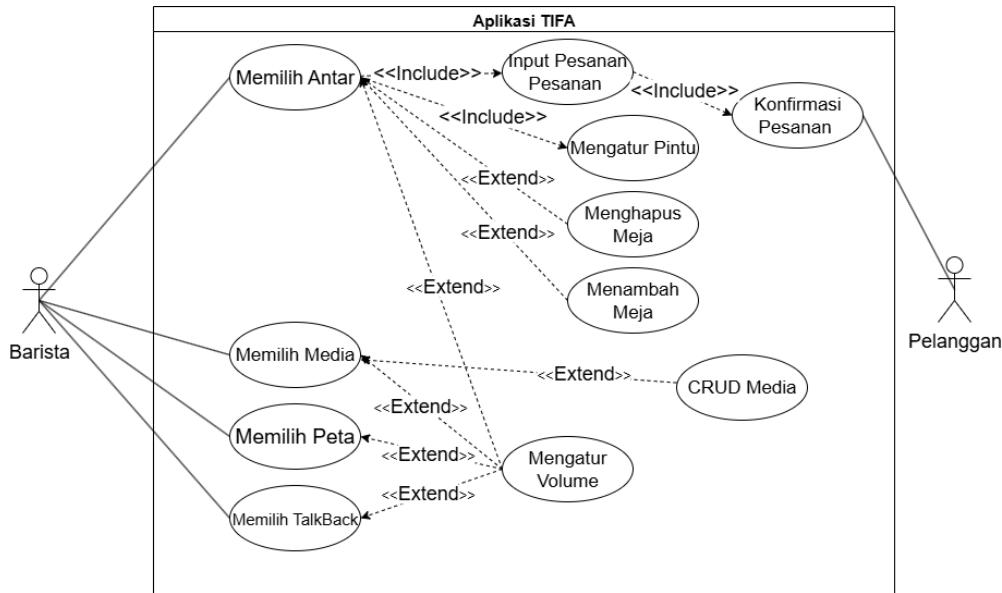
**Gambar 3. 10 Flowchart Web Monitoring**

### 3.3.6 UML (Unified Modeling Language)

UML memodelkan sistem secara terstruktur, dilengkapi *Flowchart* untuk menggambarkan alur aplikasi dan web monitoring. Rincian *Flowchart* akan dijelaskan pada sub bab berikut:

## A. Use Case Aplikasi

Gambar 3.11 menggambarkan alur interaksi antara dua aktor utama, yaitu Barista dan Pelanggan, dengan berbagai fitur inti dalam Aplikasi TIFA. Barista berperan aktif dalam mengatur layanan serta konfigurasi sistem, sedangkan Pelanggan terlibat dalam proses akhir konfirmasi pesanan.



**Gambar 3. 11 Use Case Aplikasi**

Use case Memilih Antar memiliki hubungan include dengan *Input Pesanan*, *Mengatur Pintu*, dan *Konfirmasi Pesanan*, menandakan bahwa ketiganya adalah langkah wajib. Selain itu, use case ini juga extend ke *Menghapus Meja* dan *Menambah Meja*, yang bersifat opsional. Sementara itu, Memilih Media memiliki extend ke *CRUD Media*, memungkinkan Barista mengelola media saat dibutuhkan.

Pada sisi lain, Memilih Peta dan Memilih *TalkBack* memiliki hubungan extend ke *Mengatur Volume*, sehingga volume dapat diatur saat fitur tersebut diakses. Use case *Konfirmasi Pesanan* melibatkan Barista dan Pelanggan secara langsung, memastikan adanya persetujuan dari Pelanggan jika telah sampai.

Secara keseluruhan, diagram ini menggambarkan alur kerja yang terstruktur dan fleksibel, dengan pemisahan jelas antara langkah wajib dan opsional melalui relasi include dan extend.

## B. Use Case Diagram Web Monitoring

Use case diagram pada gambar 3.12 menggambarkan berbagai aktivitas yang dapat dilakukan oleh Administrator dalam sistem pemantauan dan manajemen robot TIFA.

Administrator adalah satu-satunya aktor yang memiliki akses penuh terhadap semua fungsi dalam sistem. Diagram ini mengilustrasikan peran administrator mulai dari proses masuk ke sistem hingga melakukan pemantauan dan pengelolaan data robot.

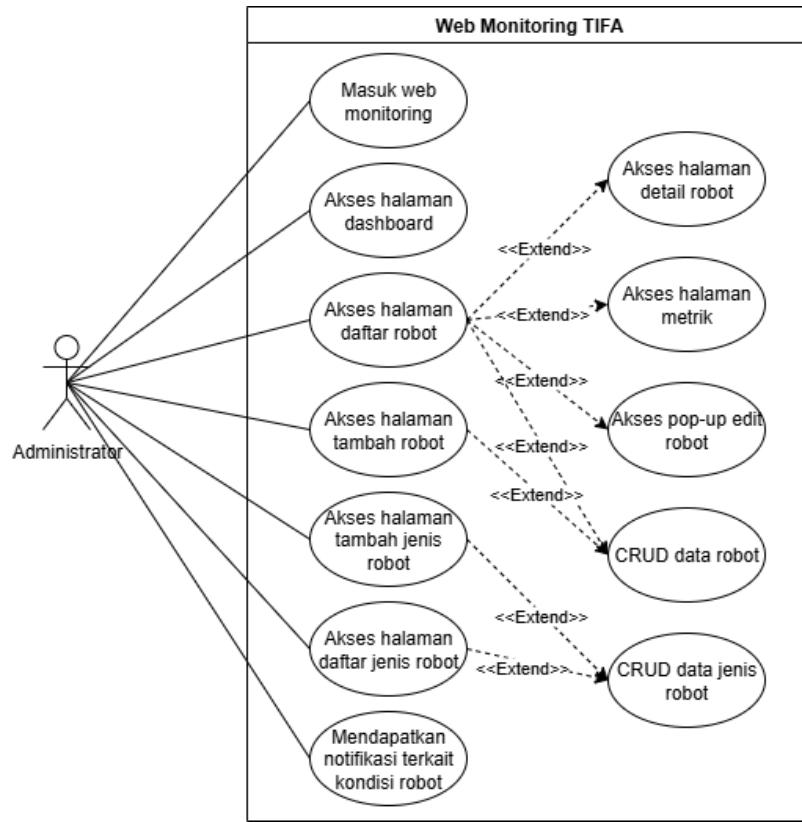
Proses dimulai dari masuk web monitoring, yaitu tahap *login* untuk mengakses sistem. Setelah berhasil *login*, administrator dapat mengakses halaman *Dashboard* yang menyajikan informasi umum terkait robot. Dari sana, administrator dapat berpindah ke halaman daftar robot, yang menampilkan seluruh robot yang telah terdaftar di sistem. Pada halaman ini, tersedia fitur tambahan yang dapat di-*extend*, seperti akses halaman Detail robot, akses halaman metrik, dan akses *Pop-up* edit robot. Ketiga fitur ini memungkinkan administrator untuk melihat informasi mendalam, statistik operasional, dan mengedit data robot secara langsung.

Selanjutnya, administrator juga dapat membuka halaman tambah robot, yang digunakan untuk menambahkan robot baru. Aksi ini terhubung dengan fitur CRUD data robot, yang merupakan bagian dari pengelolaan penuh terhadap data robot meliputi menambah, membaca, memperbarui, dan menghapus data. Fitur CRUD ini juga di-*extend* dari halaman daftar robot, karena pada halaman tersebut administrator dapat mengedit dan menghapus data robot yang sudah ada.

Selain mengelola robot, administrator juga dapat mengelola jenis robot. Diagram menunjukkan akses ke halaman tambah jenis robot dan halaman daftar jenis robot, yang keduanya terhubung ke fitur CRUD data jenis robot. Hal ini menunjukkan bahwa administrator dapat menambahkan jenis robot baru, melihat daftar jenis robot, atau menghapusnya jika tidak diperlukan lagi.

Terakhir, administrator juga dapat mendapatkan notifikasi terkait kondisi robot. Fitur ini tidak memerlukan interaksi manual, melainkan merupakan sistem pemantauan otomatis yang memberi informasi saat ada kondisi penting seperti baterai lemah atau performa baterai menurun.

Secara keseluruhan, diagram ini menunjukkan bahwa sistem web monitoring TIFA dirancang untuk mendukung administrator dalam melakukan pengawasan, pengelolaan, dan analisis robot secara komprehensif. Hubungan antar use case seperti <> menunjukkan bahwa beberapa fungsi bersifat opsional dan hanya akan dijalankan saat dibutuhkan dalam konteks penggunaan yang relevan.



**Gambar 3. 12 Use Case Diagram Web Monitoring**

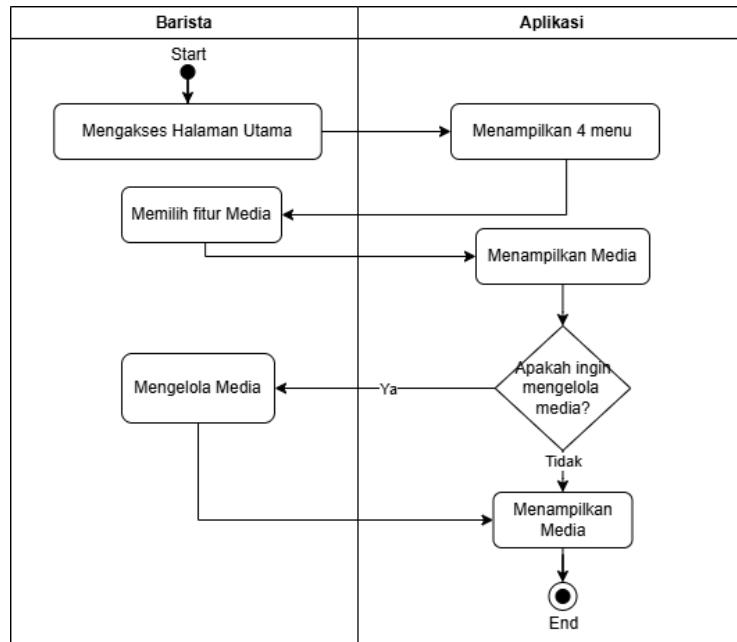
### 3.3.7 Activity Diagram

Activity diagram memvisualisasikan alur aktivitas sistem pada aplikasi dan web monitoring. Rinciannya akan dijelaskan dalam sub bab berikut.

#### A. Activity Diagram Aplikasi

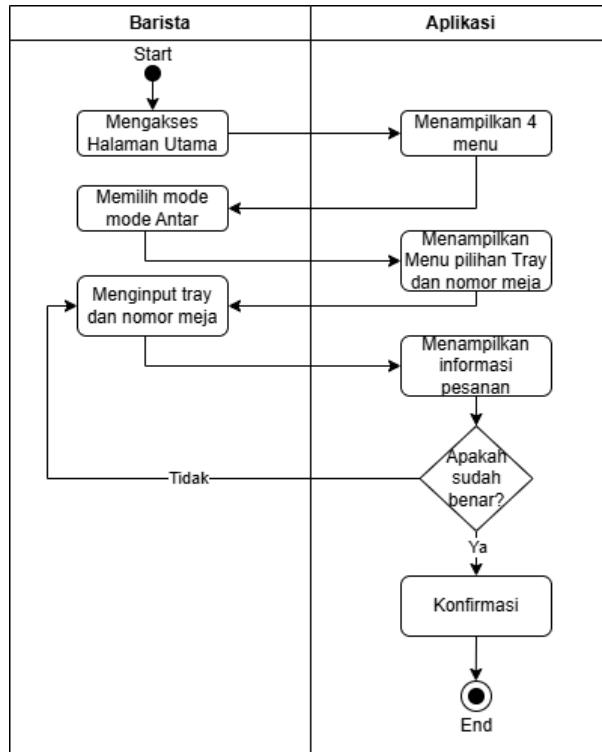
Diagram aktivitas pada gambar 3.13 menggambarkan alur interaksi antara Barista dan aplikasi saat menggunakan fitur Media. Proses dimulai ketika Barista mengakses Halaman Utama aplikasi. Setelah halaman utama berhasil diakses, aplikasi akan menampilkan empat menu utama yang tersedia. Barista kemudian memilih fitur Media dari daftar menu tersebut. Setelah fitur Media dipilih, aplikasi akan memproses dan menampilkan media (seperti foto atau video) yang tersedia. Aktivitas ini diakhiri dengan status End, menandakan bahwa proses untuk menampilkan media telah selesai.

Diagram ini memberikan gambaran yang jelas dan ringkas mengenai langkah-langkah yang dilakukan Barista dalam mengakses serta menggunakan fitur Media di aplikasi, menunjukkan interaksi sederhana namun fungsional antara pengguna dan sistem.



**Gambar 3. 13 Activity Diagram Aplikasi fitur Media**

Diagram pada gambar 3.14 ini menggambarkan alur aktivitas Barista saat menggunakan aplikasi dalam mode antar.



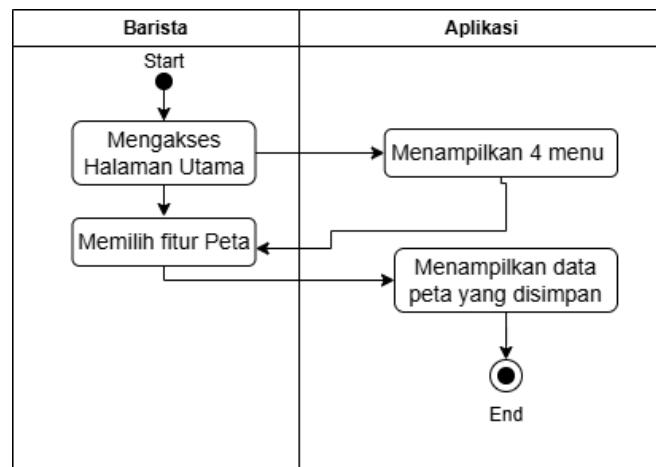
**Gambar 3. 14 Activity Diagram Aplikasi Mode Antar**

Proses dimulai dengan Barista mengakses Halaman Utama (*Start*). Setelah itu, aplikasi menampilkan empat pilihan menu kepada Barista. Barista kemudian memilih mode antar dari

opsi yang tersedia. Setelah mode antar dipilih, aplikasi menampilkan menu untuk memasukkan pilihan *tray* dan nomor meja. Barista kemudian mengkan data tersebut. Setelah informasi dimasukkan, aplikasi menampilkan Detail informasi pesanan. Pada tahap ini, Barista diminta untuk memverifikasi kebenaran informasi yang ditampilkan.

Jika informasi belum benar, Barista akan kembali ke tahap pengisian *input tray* dan nomor meja untuk memperbaiki data. Jika informasi sudah benar, Barista melakukan konfirmasi, dan proses berakhir (*End*). Diagram ini menjelaskan bagaimana aplikasi mendukung Barista dalam mengelola pesanan dengan mode antar, dengan memastikan informasi pesanan yang akurat sebelum proses konfirmasi.

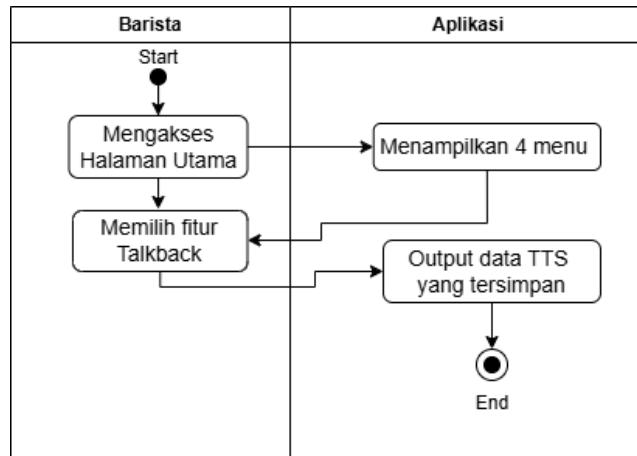
Diagram pada Gambar 3.15 menggambarkan alur aktivitas Barista saat menggunakan aplikasi untuk mengakses fitur peta. Proses dimulai ketika Barista mengakses Halaman Utama (Start). Selanjutnya, aplikasi menampilkan empat pilihan menu kepada Barista. Barista kemudian memilih fitur Peta dari menu yang tersedia. Setelah itu, aplikasi menampilkan data peta yang telah disimpan sebelumnya. Proses berakhir setelah data peta ditampilkan (End). Diagram ini memberikan gambaran ringkas mengenai interaksi antara Barista dan aplikasi dalam mengakses serta menampilkan data peta yang tersedia.



**Gambar 3. 15 Activity Diagram Aplikasi Fitur Peta**

Diagram pada Gambar 3.16 menggambarkan alur aktivitas Barista saat menggunakan aplikasi untuk mengakses fitur *TalkBack*. Proses dimulai ketika Barista membuka Halaman Utama aplikasi (Start). Setelah itu, aplikasi menampilkan empat menu utama kepada Barista. Dari pilihan yang tersedia, Barista memilih fitur *TalkBack*. Aplikasi kemudian merespons dengan menampilkan output data Text-to-Speech (TTS) yang telah disimpan sebelumnya. Setelah data *TalkBack* ditampilkan dan dijalankan oleh sistem, proses berakhir (End). Diagram

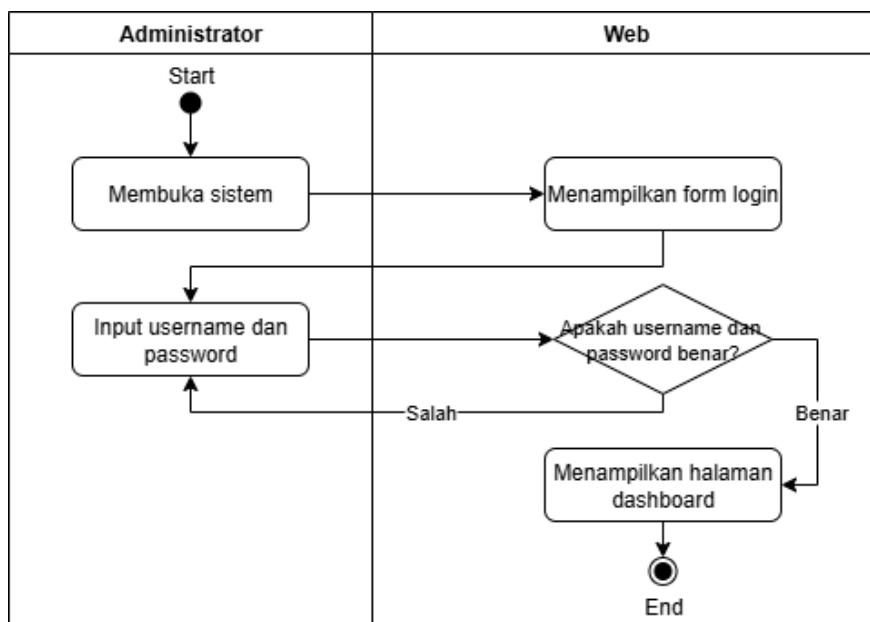
ini menggambarkan secara sederhana interaksi pengguna dengan fitur *TalkBack* yang memanfaatkan TTS untuk menyampaikan informasi secara audio kepada Barista.



**Gambar 3. 16 Activity Diagram Aplikasi Fitur *TalkBack***

### B. Activity Diagram Web Monitoring

Activity diagram yang dipaparkan pada gambar 3.17 menjelaskan aktivitas admin dalam mengelola sistem berbasis web. Proses dimulai ketika admin membuka sistem yang akan menampilkan form *login* untuk diinput *username* dan *password*. Sistem kemudian memvalidasi data yang dimasukkan. Jika data tidak valid, admin diarahkan untuk mengulang *login*. Namun, jika validasi berhasil, sistem akan menampilkan halaman *Dashboard*.



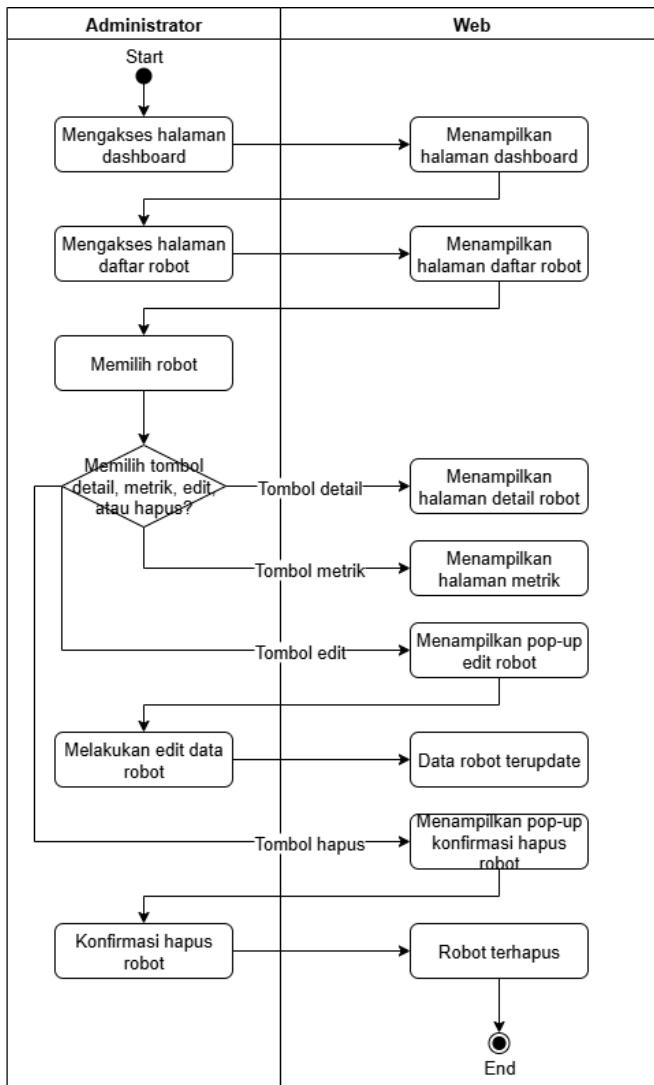
**Gambar 3. 17 Activity Diagram Web Monitoring Fitur *Login***

Diagram aktivitas di bawah pada gambar 3.18 menggambarkan alur aktivitas administrator dalam sistem web monitoring TIFA, khususnya saat melakukan pengelolaan data robot. Proses

dimulai ketika administrator mengakses halaman *Dashboard*, dan sistem web menampilkan tampilan *Dashboard* sesuai permintaan. Dari sana, administrator melanjutkan untuk mengakses halaman daftar robot, dan sistem kemudian menampilkan daftar robot yang tersedia untuk dipilih.

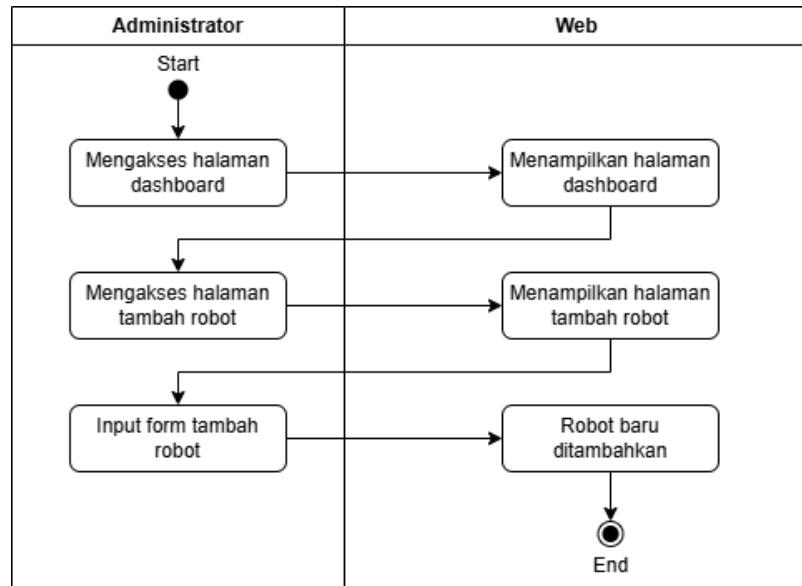
Setelah memilih salah satu robot, administrator dihadapkan pada beberapa opsi tindakan, melihat Detail robot, melihat metrik, mengedit data, atau menghapus robot. Jika administrator memilih tombol Detail, sistem akan menampilkan halaman yang berisi informasi lengkap tentang robot tersebut. Jika yang dipilih adalah tombol metrik, maka sistem akan menampilkan halaman metrik robot. Jika administrator memilih edit, maka sistem akan menampilkan *Pop-up* untuk mengedit data robot. Setelah perubahan dilakukan, sistem akan menyimpan pembaruan tersebut dan menampilkan informasi bahwa data robot telah terupdate.

Apabila administrator memilih tombol hapus, sistem akan menampilkan *Pop-up* konfirmasi. Setelah administrator mengonfirmasi tindakan ini, sistem akan menghapus data robot dari sistem dan menampilkan informasi bahwa robot telah berhasil dihapus. Seluruh alur berakhir ketika tindakan yang dipilih oleh administrator selesai dilakukan, baik berupa pembaruan data maupun penghapusan robot. Diagram ini menunjukkan bagaimana sistem *merespons* setiap aksi pengguna dengan langkah yang terstruktur dan terotomatisasi, mencerminkan integrasi yang baik antara antarmuka pengguna dan logika *backend* dalam sistem manajemen robot TIFA.



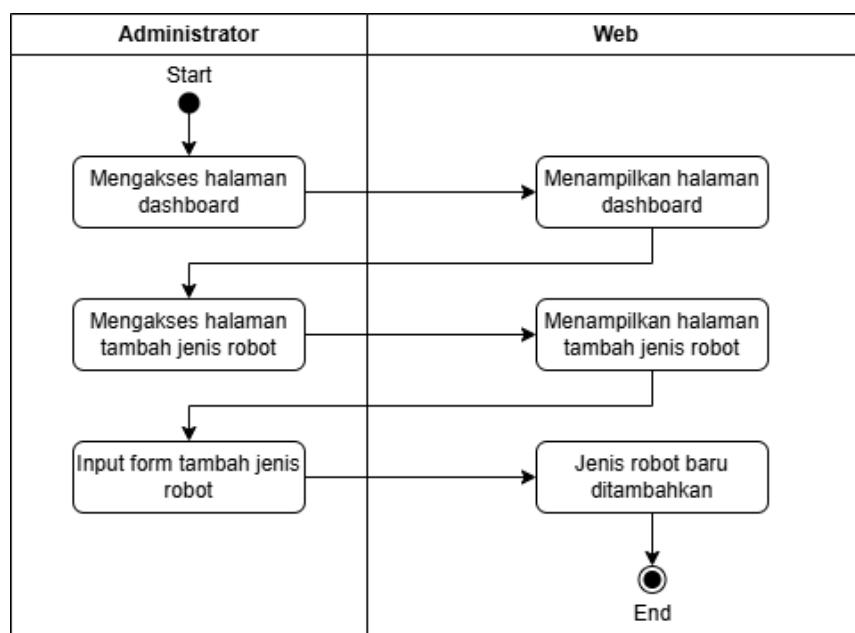
**Gambar 3.18 Activity Diagram Web Monitoring Fitur Daftar Robot**

Diagram aktivitas pada gambar 3.19 menggambarkan proses penambahan robot baru oleh administrator melalui antarmuka web. Proses dimulai ketika administrator mengakses halaman *Dashboard*, dan sistem web menampilkan halaman *Dashboard* tersebut. Selanjutnya, administrator mengakses halaman untuk menambah robot dan sistem akan menampilkan form tambah robot. Administrator kemudian mengisi form dengan data robot yang ingin ditambahkan. Setelah form diinput dan disimpan, sistem akan menyimpan data tersebut, sehingga robot baru berhasil ditambahkan ke dalam sistem. Proses ini kemudian berakhir.



**Gambar 3. 19 Activity Diagram Web Monitoring Fitur Tambah Robot**

Diagram aktivitas pada gambar 3.20 menunjukkan proses penambahan jenis robot baru oleh administrator melalui sistem berbasis web. Proses dimulai dengan administrator mengakses halaman *Dashboard*, yang kemudian akan menampilkan halaman *Dashboard* oleh sistem web. Setelah itu, administrator memilih untuk membuka halaman tambah jenis robot, dan sistem pun menampilkan halaman tersebut. Selanjutnya, administrator mengisi form tambah jenis robot dengan data yang diperlukan. Setelah form dikirim, sistem akan menyimpan data dan menambahkan jenis robot baru. Proses ini kemudian berakhir.

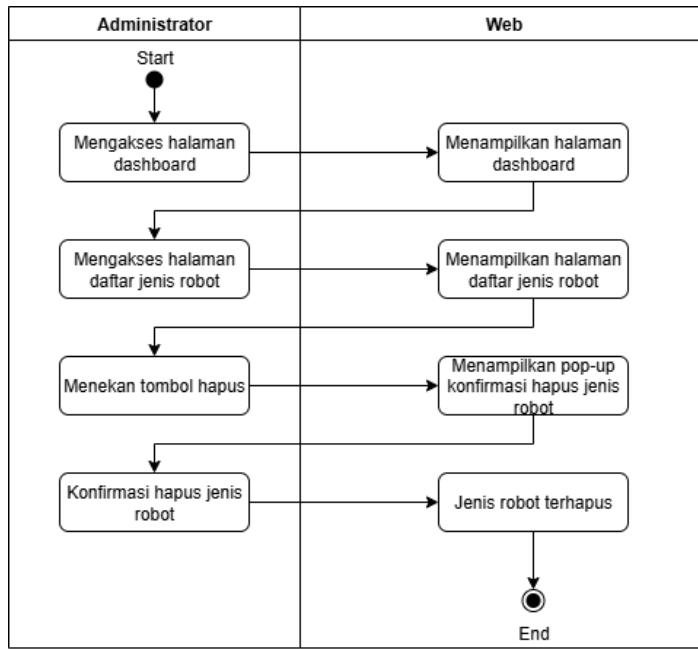


**Gambar 3. 20 Activity Diagram Web Monitoring Fitur Tambah Jenis Robot**

Activity diagram pada gambar 3.21 menjelaskan alur aktivitas administrator dalam sistem web monitoring TIFA saat melakukan penghapusan data jenis robot. Proses dimulai ketika administrator mengakses halaman *Dashboard*, dan sistem merespons dengan menampilkan tampilan *Dashboard*. Dari *Dashboard*, administrator melanjutkan ke halaman daftar jenis robot, dan sistem akan menampilkan daftar lengkap jenis-jenis robot yang sudah terdaftar di dalam sistem.

Setelah daftar jenis robot ditampilkan, administrator kemudian menekan tombol hapus pada salah satu entri jenis robot yang ingin dihapus. Sebagai respon, sistem akan menampilkan *Pop-up* konfirmasi penghapusan untuk memastikan bahwa tindakan tersebut benar-benar diinginkan. Jika administrator melanjutkan dengan konfirmasi penghapusan, maka sistem akan menjalankan proses penghapusan dan menghapus data jenis robot yang bersangkutan. Proses ini kemudian ditutup dengan status bahwa jenis robot telah berhasil dihapus, dan alur berakhir di titik *end*.

Diagram ini menggambarkan bagaimana sistem memberikan kontrol penuh kepada administrator, namun tetap menyertakan langkah verifikasi (konfirmasi) sebelum melakukan penghapusan data yang bersifat permanen. Hal ini mencerminkan prinsip keamanan dan kejelasan dalam antarmuka interaktif sistem TIFA.



**Gambar 3. 21 Activity Diagram Web Monitoring Fitur Daftar Jenis Robot**

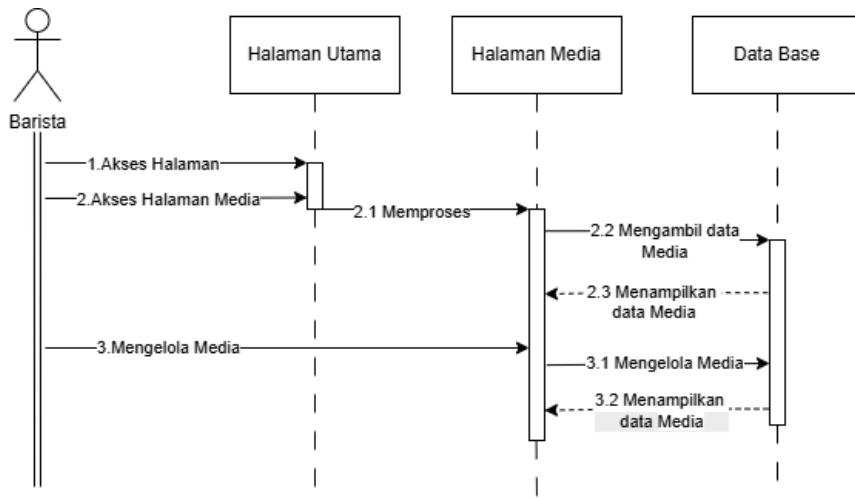
### 3.3.8 Sequence

Diagram Sequence digunakan untuk menggambarkan interaksi antar komponen dalam sistem secara berurutan. Penjelasan detil mengenai sequence pada aplikasi dan web monitoring akan diuraikan dalam sub bab berikut.

#### A. Sequence Aplikasi

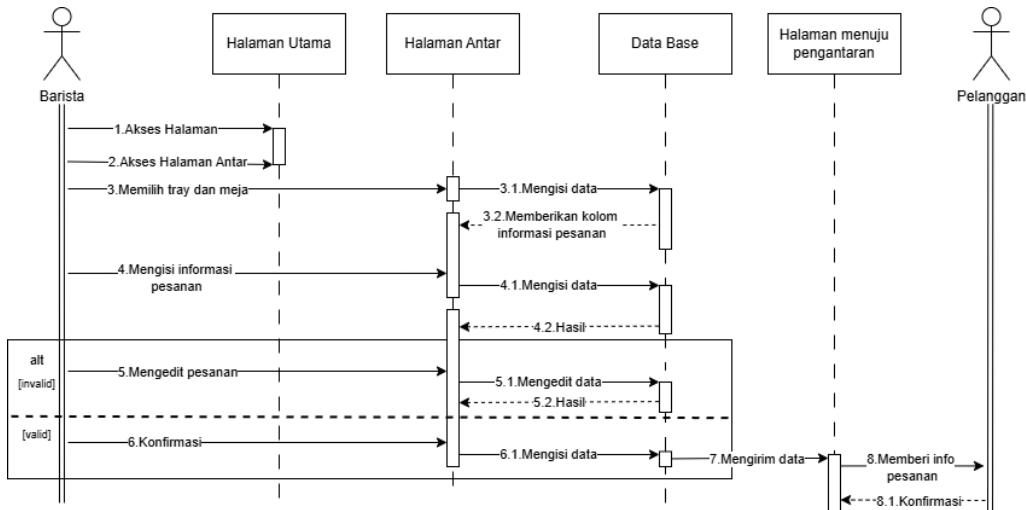
Diagram sequence pada gambar 3.22 menggambarkan interaksi antara seorang Barista dengan sistem melalui dua halaman utama, yaitu Halaman Utama dan Halaman Media. Diagram ini berfokus pada proses perpindahan antar halaman serta pengelolaan data media dalam sistem. Interaksi dimulai ketika Barista mengakses Halaman Utama (1. Akses Halaman), lalu melanjutkan dengan memilih untuk membuka Halaman Media (2. Akses Halaman Media). Permintaan ini kemudian diproses oleh sistem (2.1 Memproses), yang diteruskan ke Halaman Media untuk mengambil data dari Database (2.2 Mengambil data Media).

Setelah data berhasil diambil, Halaman Media akan menampilkan data media yang diminta (2.3 Menampilkan data Media). Selanjutnya, Barista dapat melakukan pengelolaan media (3. Mengelola Media), yang akan diproses oleh Halaman Media (3.1 Mengelola Media) dan ditampilkan kembali hasilnya (3.2 Menampilkan data Media).



**Gambar 3. 22 Sequence Aplikasi Media**

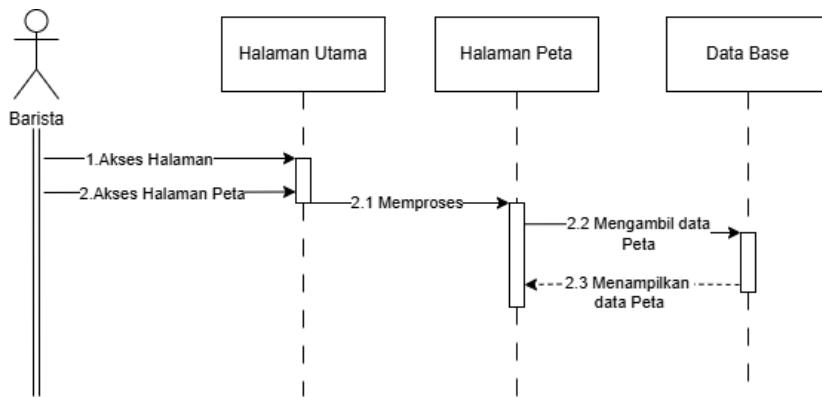
Alur ini menunjukkan urutan logis dari interaksi pengguna dengan sistem dalam mengakses dan mengelola media, serta menggambarkan komunikasi antara halaman antarmuka dan basis data yang terlibat.



**Gambar 3. 23 Sequence Aplikasi Mode Antar**

Diagram sequence yang ditunjukkan pada gambar 3.23 menjelaskan proses mode antar dalam sistem, melibatkan interaksi antara Barista, sistem, dan pelanggan. Proses dimulai dengan Barista mengakses Halaman Utama (1. Akses Halaman) dan kemudian beralih ke Halaman Antar (2. Akses Halaman Antar). Selanjutnya, Barista memilih *tray* dan *meja* yang akan digunakan (3. Memilih *tray* dan *meja*). Informasi pesanan diinputkan oleh Barista pada Halaman Antar (4. Mengisi informasi pesanan), dan data tersebut dikirimkan ke *Database* untuk diproses (4.1 Mengisi data) hingga memberikan hasil (4.2 Hasil).

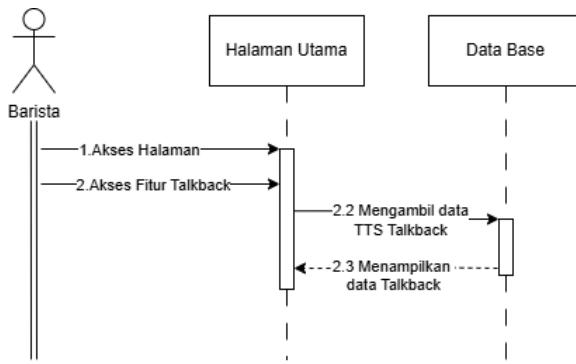
Sistem juga mencakup opsi pengeditan jika data yang dimasukkan tidak valid (5. Mengedit pesanan), dengan proses tambahan untuk mengubah informasi di Database (5.1 Mengedit data dan 5.2 Hasil). Setelah data valid, Barista melakukan konfirmasi pesanan (6. Konfirmasi), dan data dikirimkan kembali ke Database (6.1 Mengisi data). Kemudian, sistem mengirim data ke Halaman menuju pengantaran (7. Mengirim data), dan informasi pesanan diberikan kepada pelanggan (8. Memberi info pesanan), diikuti oleh konfirmasi pelanggan (8.1 Konfirmasi). Diagram ini menunjukkan alur kerja lengkap dari pembuatan pesanan hingga konfirmasi oleh pelanggan, memberikan gambaran Detail tentang sistem mode antar.



**Gambar 3. 24 Sequence Aplikasi Fitur Peta**

Diagram pada Gambar 3.24 menggambarkan alur interaksi Barista saat menggunakan mode peta dalam sistem. Proses dimulai ketika Barista mengakses Halaman Utama (1. Akses Halaman), kemudian memilih untuk membuka Halaman Peta (2. Akses Halaman Peta). Setelah itu, Halaman Utama memproses permintaan tersebut (2.1 Memproses) dan meneruskannya ke Halaman Peta. Di Halaman Peta, sistem melakukan pengambilan data peta dari Database (2.2 Mengambil data Peta). Setelah data berhasil diambil, halaman peta kemudian menampilkan data peta tersebut kepada pengguna (2.3 Menampilkan data Peta).

Alur ini menunjukkan proses sederhana namun terstruktur dari perpindahan antar halaman dan pengambilan data yang diperlukan untuk menampilkan informasi peta, sebagai bagian dari fitur mode peta dalam aplikasi.



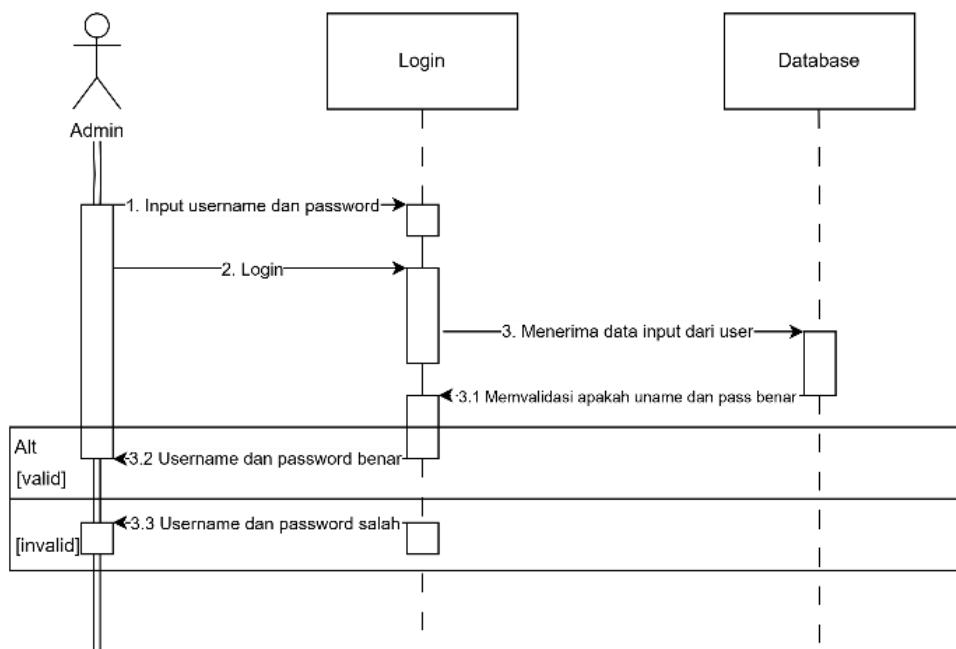
**Gambar 3. 25 Sequence Aplikasi Fitur *TalkBack***

Diagram pada Gambar 3.25 menggambarkan alur interaksi Barista saat mengakses fitur TTS (*Text-to-Speech*) *TalkBack* dalam aplikasi. Proses dimulai ketika Barista mengakses Halaman Utama (1. Akses Halaman), lalu melanjutkan dengan memilih Fitur *TalkBack* (2. Akses Fitur *TalkBack*).

Setelah itu, sistem akan melakukan pengambilan data TTS *TalkBack* dari Database (2.2 Mengambil data TTS *TalkBack*), yang kemudian hasilnya dikirimkan kembali ke Halaman Utama untuk ditampilkan kepada pengguna (2.3 Menampilkan data *TalkBack*). Alur ini menjelaskan tahapan proses akses terhadap fitur *TalkBack* yang mengandalkan data dari basis data untuk memberikan umpan balik suara kepada pengguna melalui teknologi TTS.

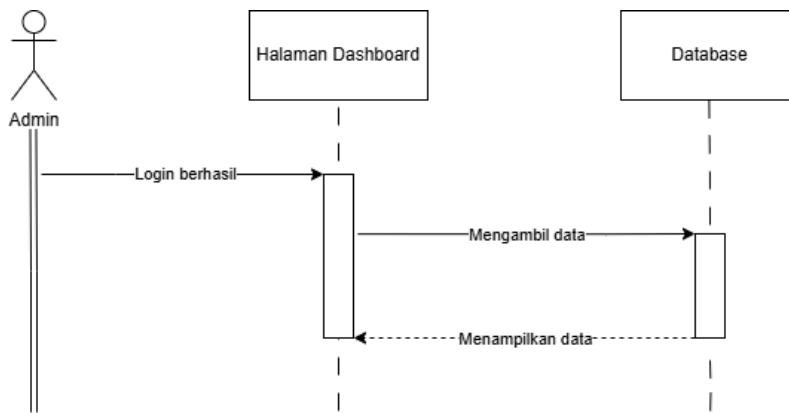
### B. Sequence Web Monitoring

*Sequence* diagram pada gambar 3.26 di bawah menggambarkan proses *login* yang melibatkan tiga entitas utama yaitu *user*, sistem *login*, dan database. Proses dimulai ketika *user* memasukkan *username* dan *password* melalui antarmuka *login*. *Input* ini kemudian dikirimkan ke sistem untuk diproses. Selanjutnya, sistem *login* menerima data tersebut dan memulai proses validasi dengan mengakses database untuk mencocokkan data *input user* dengan data yang tersimpan. Proses validasi ini menentukan apakah *username* dan *password* yang dimasukkan sesuai atau tidak. Jika data valid, sistem mengirimkan pesan balasan yang menyatakan bahwa *login* berhasil, dan *user* diberikan akses ke sistem. Sebaliknya, jika data tidak cocok, sistem akan memberikan notifikasi bahwa *username* atau *password* salah, sehingga *user* harus mengulangi proses *login* dengan data yang benar. Diagram ini menunjukkan alur komunikasi secara terstruktur untuk memastikan keamanan dan keakuratan proses autentikasi.



**Gambar 3. 26 Sequence Diagram Web Monitoring Fitur Login**

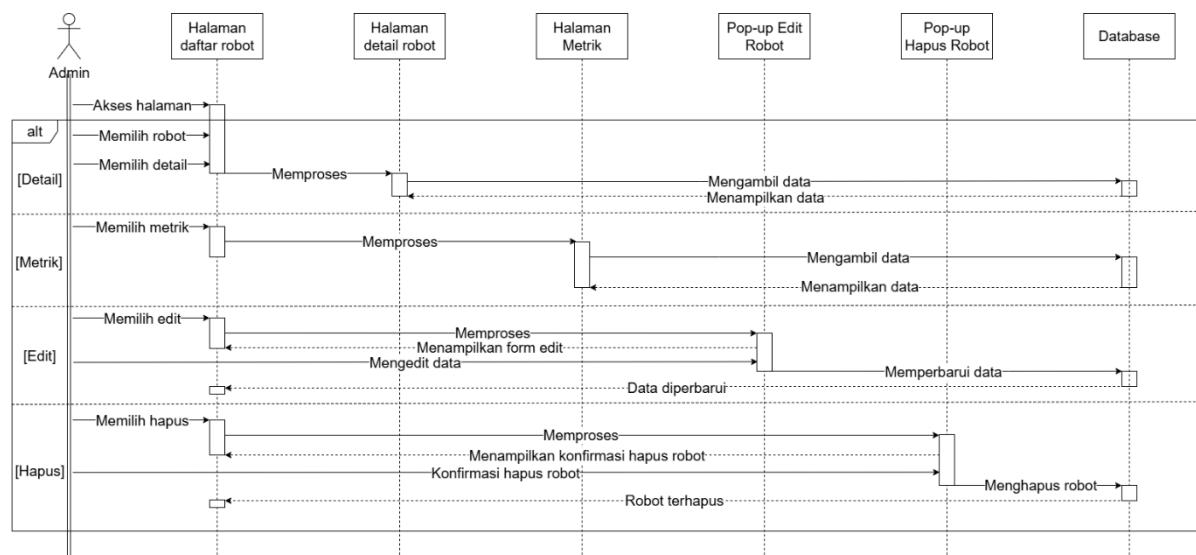
Diagram urutan (sequence diagram) pada gambar 3.27 menjelaskan alur interaksi yang terjadi setelah admin berhasil *login* ke sistem. Setelah proses *login* berhasil, admin diarahkan ke halaman *Dashboard*. Di dalam *Dashboard*, sistem secara otomatis melakukan proses mengambil data dari database, yang kemudian datanya dikirim kembali ke *Dashboard* untuk ditampilkan kepada admin. Garis panah solid menunjukkan alur komunikasi langsung (sinkron), sedangkan garis putus-putus menandakan *respons* dari proses tersebut, dalam hal ini adalah tampilan data yang telah diambil dari database.



**Gambar 3. 27 Sequence Diagram Web Monitoring Fitur Dashboard**

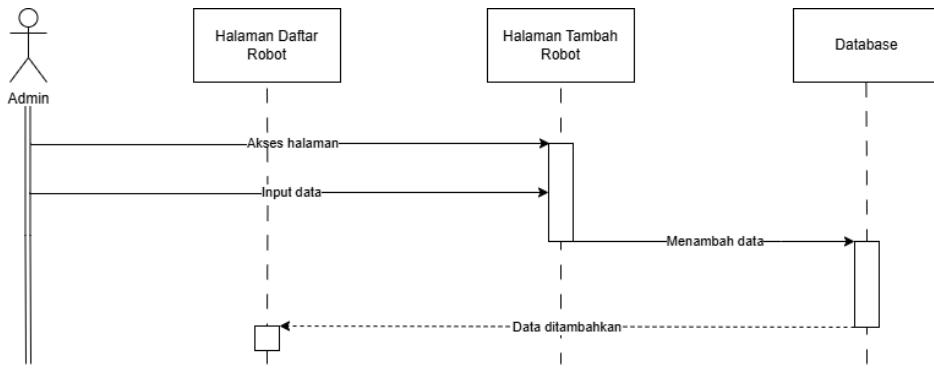
Sequence diagram pada gambar 3.28 menggambarkan interaksi antara admin dan komponen-komponen sistem web monitoring ketika mengakses dan mengelola data robot

melalui halaman daftar robot. Setelah administrator membuka halaman tersebut, ia dapat memilih salah satu robot dan menentukan tindakan lanjutan seperti melihat Detail, melihat metrik, mengedit, atau menghapus robot. Jika administrator memilih opsi Detail, sistem akan memproses permintaan tersebut dan menampilkan informasi lengkap pada halaman Detail robot. Jika yang dipilih adalah metrik, sistem akan memproses dan menampilkan grafik melalui halaman metrik. Saat memilih edit, sistem menampilkan form edit robot melalui *Pop-up*, lalu setelah admin mengirimkan perubahan, sistem akan memperbarui data di database. Untuk aksi hapus, sistem memunculkan *Pop-up* konfirmasi, dan jika admin menyetujui, data robot akan dihapus dari database. Diagram ini menunjukkan bagaimana setiap aksi yang dipicu oleh pengguna akan berinteraksi dengan halaman tertentu dan database melalui proses pengambilan, penampilan, dan pembaruan data, dengan alur komunikasi yang tertata secara berurutan dan sinkron.



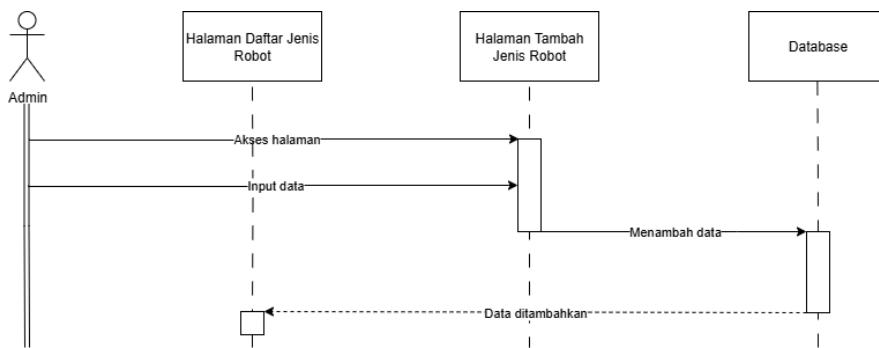
**Gambar 3. 28 Sequence Diagram Web Monitoring Fitur Daftar Robot**

Diagram urutan (sequence diagram) pada gambar 3.29 di bawah menggambarkan alur proses ketika seorang admin ingin menambahkan robot baru ke dalam sistem. Proses dimulai saat admin mengakses halaman tambah robot. Setelah halaman tersebut terbuka, admin mengisi data yang diperlukan terkait robot yang akan ditambahkan. Data yang telah *diinput* kemudian dikirim ke server untuk diproses. Server akan menyimpan data tersebut ke dalam database. Setelah proses penyimpanan berhasil, sistem akan mengirimkan *respons* kembali ke antarmuka untuk menampilkan data atau notifikasi keberhasilan. Diagram ini menunjukkan interaksi antara aktor (admin), antarmuka halaman tambah robot, dan sistem *backend* (database), menggambarkan alur logis dari permintaan hingga penyimpanan data.



**Gambar 3. 29 Sequence Diagram Web Monitoring Fitur Tambah Robot**

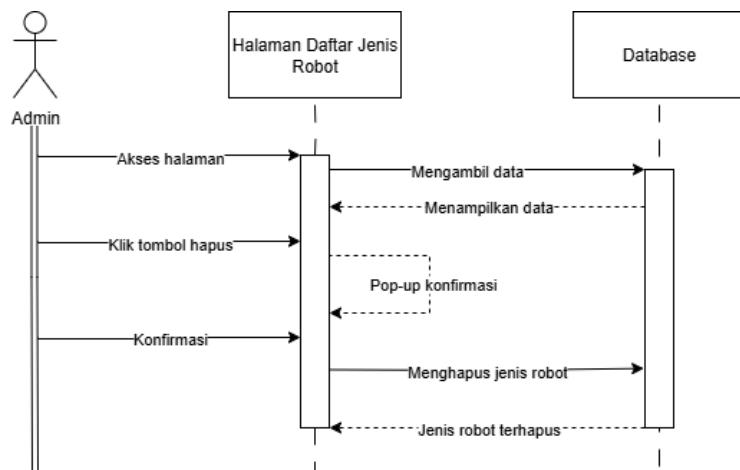
Sequence diagram pada gambar 3.30 di bawah ini menjelaskan proses penambahan jenis robot baru oleh administrator dalam sistem web monitoring. Proses dimulai ketika administrator mengakses halaman tambah jenis robot. Di halaman tersebut, administrator mengisi formulir *input*, kemudian mengirimkan data tersebut ke sistem. Selanjutnya, sistem memproses *input* tersebut dan mengirimkan permintaan untuk menambahkan data ke dalam database. Setelah data berhasil disimpan, sistem memberikan *respon*s bahwa data jenis robot telah berhasil ditambahkan, sehingga administrator dapat kembali melihatnya di halaman daftar jenis robot yang diperbarui. Diagram ini mencerminkan alur komunikasi logis dan runtut antara pengguna, antarmuka, dan *backend*.



**Gambar 3. 30 Sequence Diagram Web Monitoring Fitur Tambah Jenis Robot**

Sequence diagram pada gambar 3.31 di bawah ini menjelaskan alur interaksi ketika administrator menghapus data jenis robot melalui halaman daftar jenis robot pada sistem web monitoring. Proses dimulai ketika administrator mengakses halaman tersebut, kemudian sistem akan mengambil dan menampilkan data jenis robot dari database. Saat administrator mengklik tombol hapus pada salah satu jenis robot, sistem menampilkan *Pop-up* konfirmasi tanpa perlu mengambil data ulang, karena informasi yang ditampilkan sudah tersedia dari hasil pemuatannya awal. Jika administrator menekan tombol konfirmasi, sistem akan mengirimkan permintaan ke

database untuk menghapus jenis robot, dan setelah proses selesai, sistem akan menginformasikan bahwa jenis robot telah berhasil dihapus.



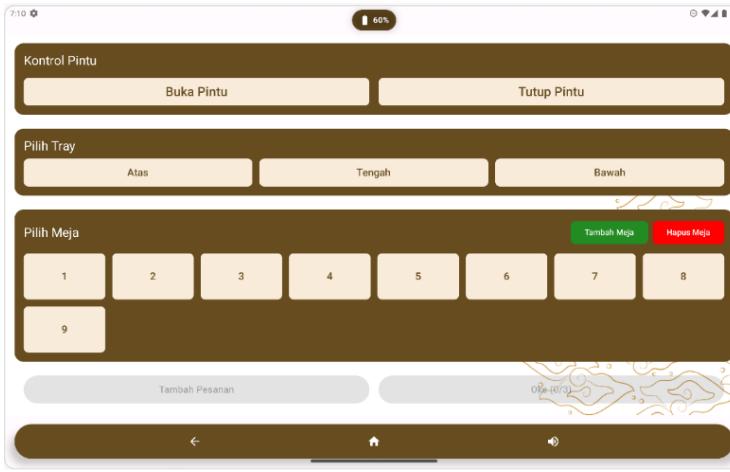
**Gambar 3. 31 Sequence Diagram Web Monitoring Fitur Daftar Jenis Robot**

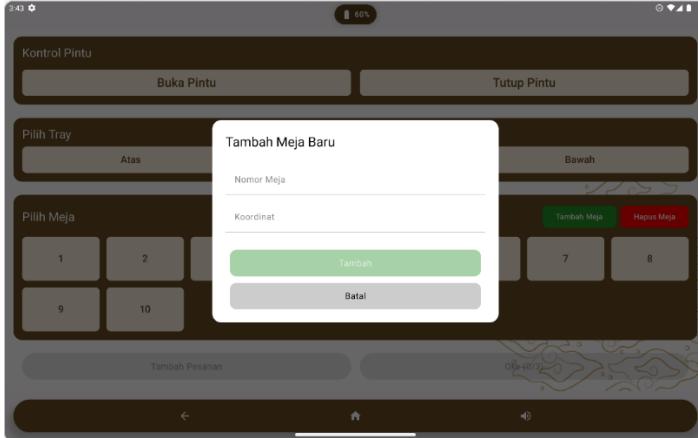
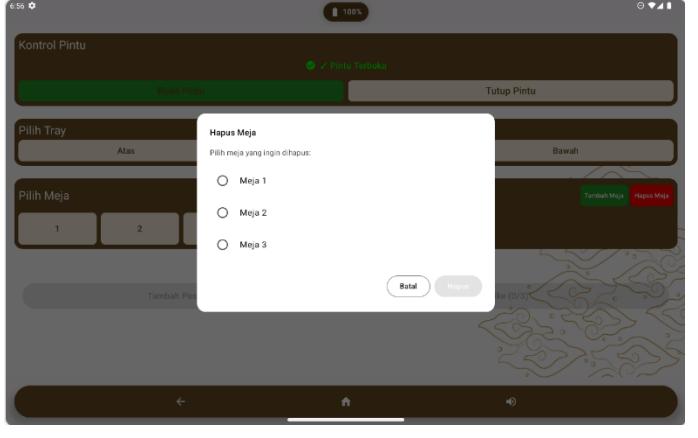
### 3.3.9 Desain UI/UX

Pada tabel 3.16 di bawah ini ditampilkan desain UI/UX setiap fitur beserta penjelasannya.

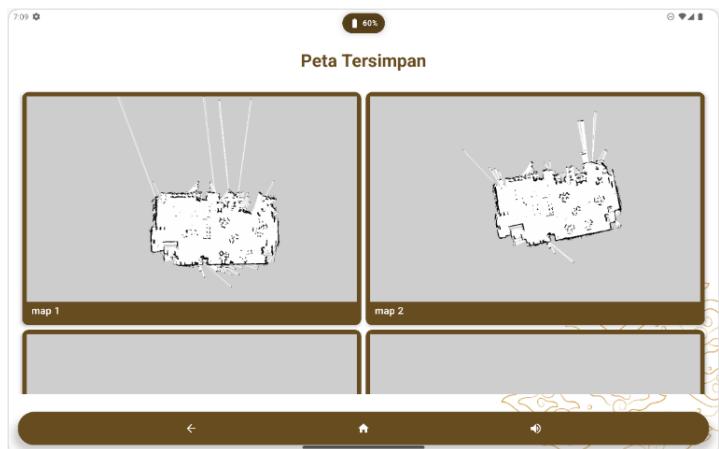
**Tabel 3.16 Desain UI/UX**

Aplikasi TIFA	
<p>Halaman Utama:</p> 	<p>Halaman utama menampilkan empat fitur yang dapat dipilih oleh pengguna, yaitu:</p> <ol style="list-style-type: none"> <li><b>Antar:</b> Fitur ini memungkinkan robot untuk mengantarkan barang ke meja tertentu yang ditentukan oleh pengguna.</li> <li><b>Media:</b> Fitur ini menampilkan konten media seperti video atau gambar yang bisa digunakan sebagai hiburan atau informasi untuk pengunjung.</li> </ol>

	<p>C. Peta: Fitur ini menampilkan peta lokasi atau area yang dapat dijelajahi oleh robot, biasanya digunakan untuk navigasi.</p> <p>D. T.I.F.A: Fitur ini menyediakan informasi mengenai sistem robot T.I.F.A itu sendiri. Setiap fitur diwakili oleh ikon dan nama yang mudah dikenali oleh pengguna. Antarmuka halaman utama ini dirancang secara sederhana dan intuitif untuk memudahkan interaksi antara pengguna dengan sistem robot.</p>
<p>Halaman Antar:</p>  <p>The screenshot shows a mobile application interface with the following sections:</p> <ul style="list-style-type: none"> <li><b>Kontrol Pintu:</b> Contains two buttons: "Buka Pintu" (Open Door) and "Tutup Pintu" (Close Door).</li> <li><b>Pilih Tray:</b> Contains three buttons: "Atas" (Top), "Tengah" (Middle), and "Bawah" (Bottom).</li> <li><b>Pilih Meja:</b> A grid of 9 numbered buttons (1-9) for selecting a table. Above the grid are two buttons: "Tambah Meja" (Add Table) in green and "Hapus Meja" (Delete Table) in red. Below the grid is a "Tambah Pesanan" (Add Order) button.</li> </ul>	<p>Pada halaman ini, pengguna dapat mengatur proses pengantaran robot dengan kontrol berikut:</p> <ol style="list-style-type: none"> <li><b>Kontrol Pintu:</b> Tombol Buka Pintu dan Tutup Pintu untuk mengoperasikan pintu <i>tray</i> secara manual.</li> <li><b>Pilih Tray:</b> Pilihan Atas, Tengah, atau Bawah untuk meletakkan barang.</li> <li><b>Pilih Meja:</b> Nomor meja (1-9) dalam grid, dengan tombol Tambah Meja (hijau) dan Hapus Meja (merah) untuk mengatur meja.</li> </ol>

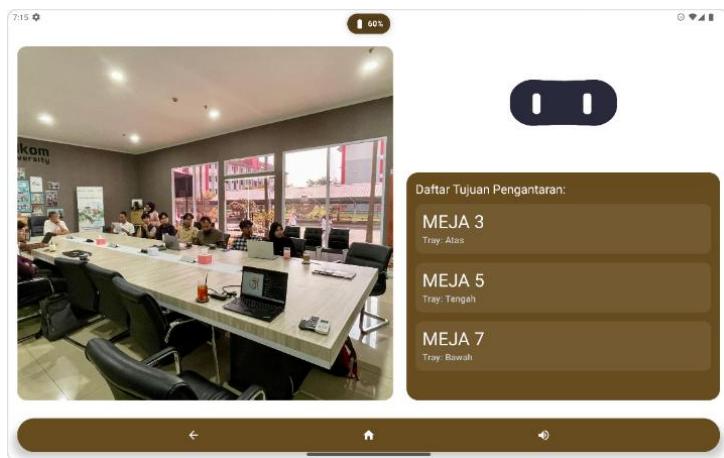
	<p>D. Bagian Bawah: Tombol Tambah Pesanan dan Oke untuk mengonfirmasi pengantaran.</p> <p>E. Antarmuka ini memudahkan pengguna mengendalikan pintu, tray, dan tujuan meja secara penuh.</p>
Tambah Meja:	<p>Pop up ini berfungsi untuk menambahkan nomor meja dengan menambahkan Nomor Meja dan Koordinat</p> 
Hapus Meja:	<p>Pop up ini berfungsi untuk menghapus nomor meja dengan memilih Nomor Meja</p> 

#### Halaman Peta:



Halaman *Maps* dirancang menampilkan daftar peta hasil mapping yang telah disimpan. Antarmuka ini memudahkan pemilihan lokasi secara visual dan cepat.

#### Halaman Informasi Pengantaran:

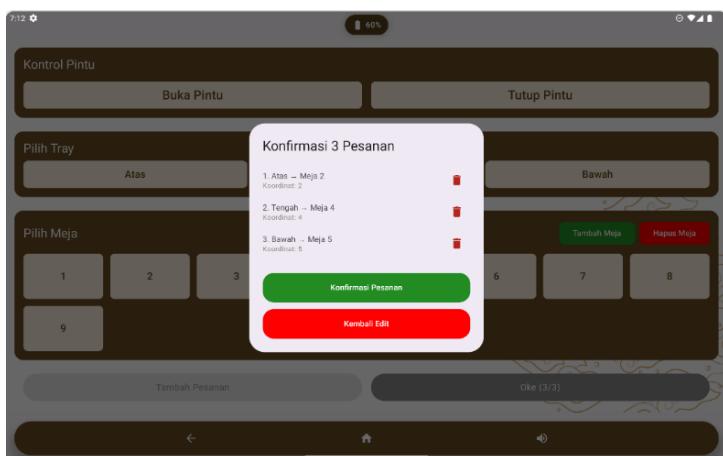


Halaman ini menampilkan Detail proses pengantaran robot dengan elemen utama:

- A. Nomor Meja: Menandai tujuan pengantaran.
- B. Lokasi *Tray*: Menunjukkan posisi barang di *tray*.
- C. Visualisasi: Terdapat animasi robot dan Media dari fitur Media yang dapat diatur.

Tampilan ini memudahkan pengguna memantau proses pengantaran secara *real-time*.

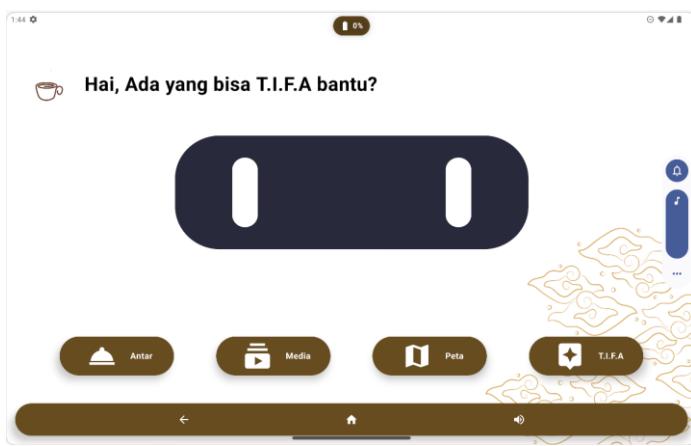
*Pop-up* Informasi Pesanan:



*Pop-up* ini menampilkan Detail pesanan untuk memastikan keakuratan pesanan 1, 2, dan 3:

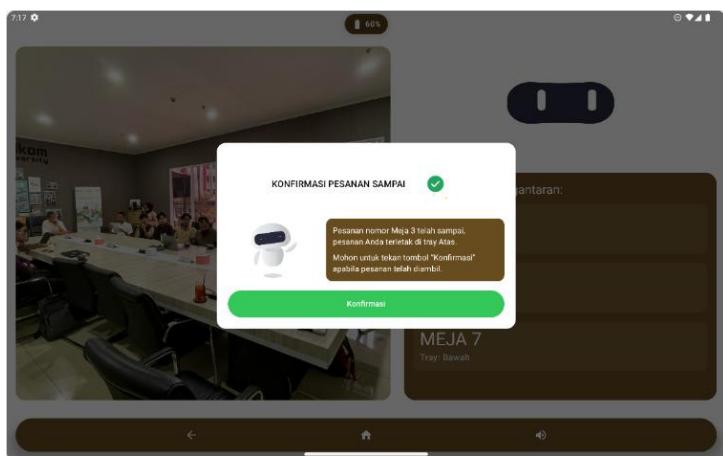
- A. Posisi *Tray*: Menunjukkan lokasi barang di *tray* (Atas, Tengah, Bawah).
- B. Nomor Meja: Menampilkan nomor meja tujuan untuk pesanan.
- C. Ikon Hapus: Mengubah informasi pesanan sesuai kebutuhan.
- D. Tombol Konfirmasi Pengantaran: Memulai proses pengantaran setelah penyesuaian.

*Pop-up* Volume:



Di sisi kanan layar, terdapat kontrol *Pop-up* untuk mengatur volume suara robot. Fitur ini membantu pengguna menyesuaikan tingkat suara sesuai kebutuhan tanpa mengganggu pengaturan lainnya.

*Pop-up Konfirmasi Pengantaran:*



*Pop-up* ini menampilkan informasi pengantaran dan konfirmasi:

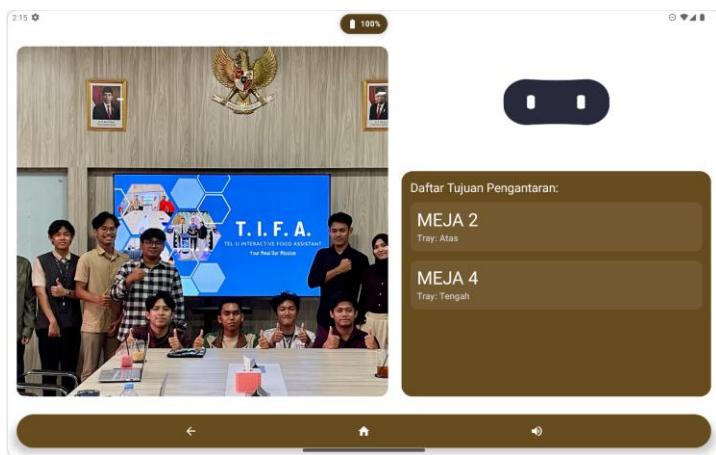
A. Pesan Konfirmasi:

Memberitahukan bahwa pesanan telah sampai sesuai dengan informasi, Pengguna diminta menekan tombol Konfirmasi setelah mengambil pesanan.

B. Tombol Konfirmasi:

bertuliskan Konfirmasi untuk menyelesaikan proses pengantaran dan mengonfirmasi bahwa pesanan telah diterima.

*Halaman Informasi Pengantaran (Pesanan Selanjutnya):*

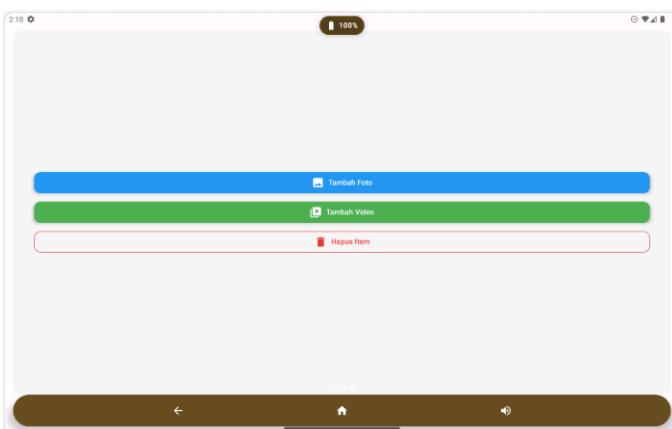


Halaman ini menampilkan pengantaran berikutnya dengan elemen utama:

A. Nomor Meja: Menunjukkan tujuan berikutnya.

B. Posisi *Tray*: Menampilkan posisi *tray* pesanan.

Halaman Media:



Halaman ini memungkinkan pengguna mengelola konten media dengan fitur berikut:

- A. Tambah Foto: Tombol biru untuk menambahkan foto ke dalam aplikasi.
- B. Tambah Video: Tombol hijau yang digunakan untuk menambahkan video.
- C. Hapus Item: Tombol merah untuk menghapus media yang telah dipilih dari daftar.

Antarmuka ini dirancang agar pengguna dapat menyesuaikan konten visual robot secara mudah dan cepat.

## Web Monitoring

Halaman *Login*:

Pada formulir *login* terdapat dua *input* utama, yaitu *input username* dan *password* yang digunakan untuk autentikasi. Tombol “*Login*” ditempatkan di bagian bawah untuk memproses akses ke sistem setelah data dimasukkan.

Halaman *Dashboard*:



*Dashboard* menampilkan ringkasan informasi utama dari semua robot yang terdaftar. Di sini, pengguna dapat melihat status baterai, performa baterai, durasi operasional robot (dalam hari), log aktivitas terbaru, serta grafik visual terkait aktivitas harian dan jumlah pesanan. Informasi ini membantu administrator memantau kinerja dan penggunaan robot secara *real-time*.

### Halaman Daftar Robot:

The screenshot shows the 'Daftar Robot' section of the TIFA application. It lists four robots under two categories: 'Delivery Robot' and 'Collaborative Robot (Cobot)'. Each robot card includes a small icon, the robot's name, its serial number, a battery level indicator, and a performance bar. Below each card are four buttons: 'Detail', 'Metrik', 'Edit', and 'Hapus'.

Halaman ini menampilkan daftar robot yang dikelompokkan berdasarkan jenisnya. Setiap robot ditampilkan dengan informasi lengkap, termasuk nama, nomor serial, persentase baterai, dan performa baterainya. Selain itu, tersedia tombol aksi seperti Detail, Metrik, Edit, dan Hapus untuk memudahkan pengguna dalam memantau dan mengelola robot. Tampilan ini dirancang agar pengguna dapat mengelola banyak robot secara efisien dalam satu sistem yang terorganisir.

### Pop-up Notifikasi:

The screenshot shows a 'Dashboard' view with a pop-up notification for a robot named 'Tifa (Serial: SN01)'. The notification states: 'Performa Baterai Perlu Perhatian' (Battery performance needs attention) and 'Baterai Lemah' (Low battery). It advises taking action immediately. The background dashboard shows other robot status cards and a chart of daily activity.

Bagian ini penting untuk memberikan notifikasi terkait kondisi robot seperti baterai robot lemah atau pemberitahuan penting lainnya.

### Halaman Detail Robot:

The screenshot shows the 'Detail Robot' page for 'Tifa (SN01)'. It features three main sections: a summary card with battery (20%), performance (89%), and operational time (3 Hari); a 'Galeri Peta Robot (Delivery Robot)' section showing four map thumbnails; and a 'Log Aktivitas Robot' section showing a timeline of activity logs.

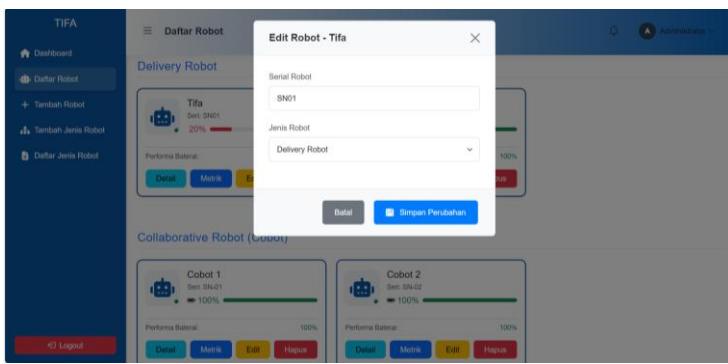
Pada halaman Detail, pengguna dapat melihat informasi spesifik dari satu robot, seperti persentase baterai, performa baterai, dan lama operasional. Selain itu, terdapat galeri peta robot (*maps*) serta log aktivitas lengkap yang menunjukkan riwayat tugas.

### Halaman Metrik:



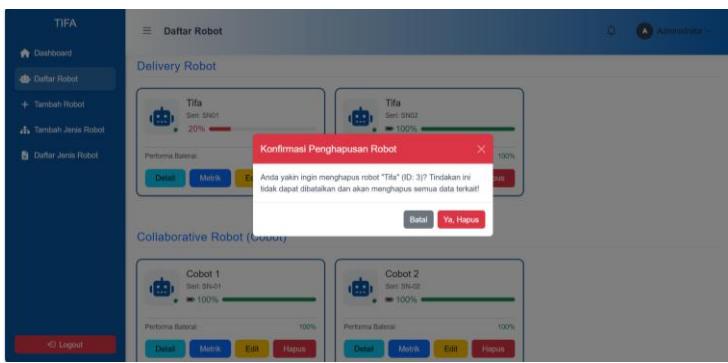
Halaman ini merupakan tampilan apabila pengguna memilih tombol “Metrik” pada halaman daftar robot. Pada halaman ini menampilkan grafik performa dan statistik robot yang telah dipilih sebelumnya.

### Pop-up Edit Robot:



*Pop-up edit robot* memungkinkan pengguna untuk memperbarui data robot, seperti nomor seri dan jenis robot. Antarmuka ini memudahkan administrator untuk melakukan pembaruan data tanpa perlu intervensi teknis langsung pada database.

### Pop-up Konfirmasi Hapus Robot:



Saat pengguna memilih untuk menghapus robot, sistem menampilkan *Pop-up* konfirmasi untuk memastikan tindakan tidak dilakukan secara tidak sengaja. Penghapusan bersifat permanen dan akan menghapus seluruh data terkait dengan robot tersebut.

#### Halaman Tambah Robot:



Halaman tambah robot memungkinkan administrator untuk mendaftarkan robot baru ke dalam sistem. Formulir ini mencakup tiga *field* utama yaitu Nama Robot, Nomor Seri, dan Jenis Robot, yang harus diisi sebelum data disimpan. Administrator dapat memilih jenis robot dari menu *dropdown*.

#### Halaman Tambah Jenis Robot:



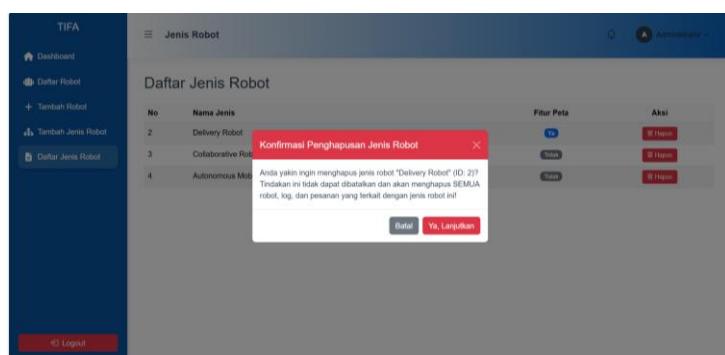
Fitur ini digunakan untuk menambahkan tipe robot baru ke sistem, misalnya untuk membedakan antara robot yang menggunakan peta atau tidak. Pengguna cukup mengisi nama jenis robot dan memilih apakah jenis tersebut memiliki fitur peta (*maps*).

#### Halaman Daftar Jenis Robot:

No	Nama Jenis	Fitur Peta	Aksi
2	Delivery Robot	<input checked="" type="checkbox"/>	<input type="button" value="Hapus"/>
3	Collaborative Robot (Cobot)	<input type="checkbox"/>	<input type="button" value="Hapus"/>
4	Autonomous Mobile Robot (AMR)	<input type="checkbox"/>	<input type="button" value="Hapus"/>

Daftar ini menampilkan seluruh jenis robot yang terdaftar beserta status apakah mereka mendukung fitur peta atau tidak. Setiap entri dapat dihapus oleh administrator bila sudah tidak relevan. Fitur ini menjaga konsistensi dan fleksibilitas klasifikasi robot.

### Pop-up Konfirmasi Hapus Jenis Robot:



Saat pengguna memilih untuk menghapus jenis robot, sistem menampilkan *Pop-up* konfirmasi untuk memastikan tindakan tidak dilakukan secara tidak sengaja. Penghapusan bersifat permanen dan akan menghapus seluruh data terkait dengan jenis robot tersebut.

## 3.4 Jadwal dan Anggaran

Bagian ini membahas perencanaan jadwal dan alokasi anggaran yang diperlukan untuk pelaksanaan proyek. Jadwal disusun untuk memastikan setiap tahapan dapat diselesaikan tepat waktu, sementara anggaran dirancang agar penggunaan sumber daya efektif dan efisien. Rincian mengenai jadwal dan anggaran akan diuraikan dalam sub bab berikut.

### 3.4.1 Jadwal

Pada tabel 3.17 dipaparkan mengenai rencana jadwal pengerjaan yang meliputi tahapan utama, sub tahapan utama, rentang kerja, durasi kerja, milestone, dan lead.

**Tabel 3. 15 Jadwal Pengerjaan**

No.	Tahapan Utama	Sub Tahapan Utama	Rentang Kerja	Durasi Kerja	Milestone	Lead
1.	Analisis Kebutuhan	Identifikasi kebutuhan <i>Hardware</i> , <i>Software</i> , dan database	02 - 15 Januari 2025	13 Hari	Dokumen kebutuhan proyek final	Kelompok
		Survei dan interview dengan mitra	02 - 10 Januari 2025	9 Hari		Figo, Nabila

No.	Tahapan Utama	Sub Tahapan Utama	Rentang Kerja	Durasi Kerja	Milestone	Lead
1.	Analisis Kebutuhan	Finalisasi kebutuhan proyek	16 - 31 Januari 2025	15 Hari	Spesifikasi teknis disetujui	Kelompok
2.	Perancangan Perangkat Keras	Pemasangan ESP32	01 Februari - 05 Februari 2025	5 Hari	Semua integrasi <i>Hardware</i> selesai dan siap dapat mengirim data.	Jibran
		Integrasi ESP32 dengan Baterai	16 Februari - 20 Februari 2025	5 Hari		Jibran
3.	Pengembangan Database	Pembuatan Database dan <i>Querynya</i>	26 Februari - 05 Maret 2025	8 Hari	Database terintegrasi sepenuhnya dengan <i>Hardware</i> .	Figo
		Pengintegrasian dengan Perangkat Keras	06 Maret - 20 Maret 2025	15 Hari		Figo, Jibran
4.	Pengembangan Aplikasi	Desain Antarmuka Aplikasi	01 Februari - 15 Februari 2025	14 Hari	Aplikasi selesai dan siap untuk pengujian.	Figo
		Pengembangan <i>Frontend</i> Aplikasi	16 Februari - 20 Maret 2025	33 Hari		Figo

No.	Tahapan Utama	Sub Tahapan Utama	Rentang Kerja	Durasi Kerja	Milestone	Lead
4.	Pengembangan Aplikasi	Pengembangan <i>Backend</i> Aplikasi	20 Maret - 05 April 2025	16 Hari	Aplikasi selesai dan siap untuk pengujian.	Figo
		Implementasi <i>Frontend</i> dan <i>Backend</i>	05 April - 14 April 2025	9 Hari		Figo
5.	Pengembangan Web	Desain Antarmuka Web	01 Februari - 15 Februari 2025	14 Hari	Web selesai dan siap untuk pengujian.	Nabila
		Pengembangan <i>Frontend</i> Web	16 Februari - 20 Maret 2025	33 Hari		Nabila
5.	Pengembangan Web	Pengembangan <i>Backend</i> Web	20 Maret - 05 April 2025	16 Hari	Web selesai dan siap untuk pengujian.	Nabila, Jibran
		Implementasi <i>Frontend</i> dan <i>Backend</i>	05 April - 14 April 2025	9 Hari		Nabila
6.	Integrasi Data	Integrasi Aplikasi dan Web	14 April - 18 April 2025	5 Hari	Semua integrasi selesai dan siap	Figo, Nabila
		Integrasi dengan Raspi dari Motion System	06 Februari - 10 Februari 2025	5 Hari	untuk mengirim dan menerima data	Figo

No.	Tahapan Utama	Sub Tahapan Utama	Rentang Kerja	Durasi Kerja	Milestone	Lead
6.	Integrasi Data	Integrasi dengan Raspi dari Sensor Infrared	21 Februari - 25 Februari 2025	4 Hari	Semua integrasi selesai dan siap untuk mengirim dan menerima data	Figo
		Integrasi dengan raspi dari ESP Baterai	26 - 27 Februari 2025	2 Hari		Figo
7.	Pengujian dan Validasi	Pengujian dan Perbaikan	15 April – 30 Juni 2025	77 Hari	Semua fitur lolos pengujian tanpa bug.	Kelompok
		Validasi	01 Juli – 07 Juli 2025	5 Hari		Kelompok

### 3.4.2 Anggaran

Pada tabel 3.18 dipaparkan mengenai anggaran yang diperlukan untuk Pengembangan Sistem Monitoring Tel-U Interactive Food Assistant (TIFA) dalam Upaya Peningkatan Pelayanan di Tel-U Coffee.

**Tabel 3. 16 Anggaran**

No.	Nama Barang	Keterangan	Volume	Harga per Unit	Total Harga
1.	ESP32S3 + Kabel Micro USB	Mikrokontroller	1 Unit	Rp. 100.000	Rp. 100.000
2.	Sensor Infrared	Sensor pendekripsi makanan	5 Unit	Rp. 10.000	Rp. 50.000
3.	<i>Voltage sensor</i>	Sensor tegangan	1 Unit	Rp. 20.000	Rp. 20.000

No.	Nama Barang	Keterangan	Volume	Harga per Unit	Total Harga
4.	Kabel Konektor <i>Micro JST</i>	Kabel Standar Penghubung untuk IoT	100 Buah	Rp. 1.500	Rp. 150.000
5.	Redmi TAB SE	Perangkat	1 Unit	Rp. 2.000.000	Rp. 2.000.000
6.	Android Studio	<i>Framework pengembangan aplikasi android</i>	1 Lisensi	Rp. 0 (Open Source)	Rp. 0
7.	Vue.js	<i>Framework Frontend web monitoring</i>	1 Lisensi	Rp. 0 (Open Source)	Rp. 0
8.	Node.js	<i>Framework backend</i>	1 Lisensi	Rp. 0 (Open Source)	Rp. 0
9.	MySQL	Database	1 Lisensi	Rp. 0 (Open Source)	Rp. 0
10.	Lightsail AWS, Domain, DNS <i>Cloud flare</i>	4 Instance (PHP BE, Express.js BE, MySQL DB, Vue.js FE) 1vCPU, 1GB RAM, 25GB SSD, Ubuntu 22.04 LTS	30 Hari	Rp. 400.000	Rp. 400.000
Total Rencana Anggaran Biaya					Rp. 2.720.000

## BAB 4

# IMPLEMENTASI

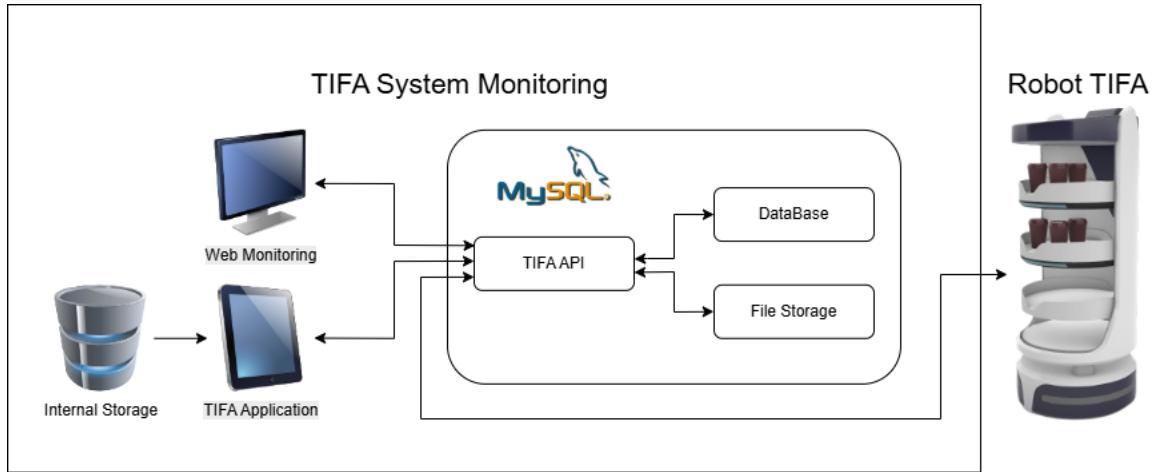
### 4.1 Deskripsi Umum Implementasi

Pada tahap implementasi, solusi yang dirancang dalam proyek *Tel-U Interactive Food Assistant* (TIFA) direalisasikan dengan sistem robot pengantar makanan berbasis *Internet of Things* (IoT) yang terintegrasi dengan aplikasi *mobile* dan platform web monitoring. Implementasi ini bertujuan untuk meningkatkan efisiensi layanan di Tel-U Coffee, khususnya dalam menghadapi lonjakan permintaan pada jam sibuk. Solusi yang diimplementasikan merupakan kombinasi dari komponen perangkat keras dan perangkat lunak yang bekerja secara sinergis. Komponen perangkat keras terdiri atas mikrokontroler ESP32-S3 untuk komunikasi nirkabel dan integrasi data, Teensy 4.1 sebagai pusat kendali utama, Arduino Mega untuk pengaturan gerakan *tray* dan pintu, serta Raspberry Pi yang memproses data dari sensor LiDAR [45], [46]. Sistem juga dilengkapi dengan sensor *Infrared* untuk mendeteksi keberadaan makanan pada *tray*.[47].

Pada sisi perangkat lunak, aplikasi *mobile* dibangun menggunakan Android Studio dan dioperasikan melalui tablet (Redmi Tab SE) sebagai antarmuka utama bagi *user*. Aplikasi ini menyediakan mode kerja, yaitu antar serta beberapa fitur tambahan seperti media, dan peta. Aplikasi ini juga dilengkapi animasi dari lottiefiles untuk memperhalus interaksi, memberikan umpan balik visual, dan memudahkan navigasi pengguna. Selain membuat aplikasi lebih menarik dan dinamis, animasi juga membantu menyampaikan informasi secara intuitif [48]. Selain itu, web monitoring dikembangkan menggunakan *Framework Vue.js* untuk *Frontend*. Sistem ini memungkinkan pemantauan status robot secara *real time*, termasuk status baterai [49], [50]. Penyimpanan dan pengelolaan data dilakukan melalui basis data MySQL yang mendukung operasi *create, read, update, delete* (CRUD) dan relasi antar entitas seperti pesanan, *tray*, dan posisi meja [7], [8].

Proses implementasi dilakukan secara kolaboratif oleh tim multidisiplin yang terdiri dari mahasiswa dari berbagai program studi, masing-masing bertanggung jawab atas sub-komponen sistem. Setiap tahapan implementasi juga didukung oleh dokumentasi teknis yang merinci alur komunikasi antar perangkat, serta pengujian sistem melalui metode *User Acceptance Testing* (UAT) untuk menjamin bahwa setiap fitur berjalan sesuai dengan kebutuhan operasional di Tel-U Coffee. Dengan pendekatan ini, sistem TIFA berhasil

diwujudkan sebagai prototipe fungsional yang siap mendukung proses layanan secara otomatis, cepat, dan efisien di lingkungan kafe.



**Gambar 4. 1 Arsitektur sistem TIFA**

Arsitektur sistem TIFA Monitoring dirancang dengan pendekatan *client-server* yang mengintegrasikan tiga komponen utama, yaitu aplikasi TIFA pada perangkat tablet, antarmuka web monitoring, dan layanan *backend* melalui TIFA API. Aplikasi TIFA digunakan oleh barista untuk mengoperasikan robot pengantar makanan dan menyimpan preferensi media lokal pada memori internal, sementara web monitoring memungkinkan admin memantau status robot secara *real-time* melalui *browser*. Kedua antarmuka ini terhubung ke TIFA API yang berfungsi sebagai pusat pengelolaan data, menangani permintaan dari pengguna, serta mengakses sistem basis data MySQL dan file *storage* untuk menyimpan data operasional seperti status baterai, log sensor, dan histori pengantaran [7].

#### 4.1.1 Pengembangan Aplikasi TIFA

Aplikasi TIFA (Tel-U Interactive Food Assistant) dikembangkan sebagai salah satu komponen utama dalam sistem layanan otomatis yang dirancang untuk mendukung operasional robot pengantar pesanan di lingkungan kafe berbasis teknologi. Aplikasi ini berfungsi sebagai kontrol robot, visualisasi media, visualisasi peta dan antarmuka utama antara operator (barista dan owner) dengan sistem.

#### 4.1.2 Pengembangan Web Monitoring

Web monitoring merupakan salah satu komponen yang penting dalam sistem layanan otomatis yang dikembangkan untuk memonitoring kondisi dan aktivitas robot. Web ini juga dapat memberikan notifikasi pada kondisi-kondisi tertentu seperti ketika baterai robot lemah.

## 4.2 Detail Implementasi

Sub bab ini menjelaskan secara rinci mengenai implementasi fitur dalam sistem monitoring TIFA. Penjelasan mencakup struktur arsitektur sistem, alur kerja, serta mekanisme komunikasi antar lapisan yang memungkinkan proses distribusi pesanan dapat berjalan secara efisien dan terkoordinasi dari perangkat pengirim hingga pengguna tujuan.

### 4.2.1 Detil Implementasi Aplikasi TIFA

Mode antar pada aplikasi TIFA dirancang untuk mengelola proses distribusi pesanan dari perangkat ke tujuan pengguna melalui sistem pemilihan *tray* dan koordinat meja. Fitur ini melibatkan arsitektur server yang terdiri dari lapisan presentasi (UI Android), logika bisnis (ViewModel dan Repository), dan lapisan *backend* (API berbasis PHP dan basis data MySQL). Setiap bagian saling berinteraksi untuk menjalankan proses pemilihan *tray*, pengambilan data meja, serta penyimpanan dan pengambilan informasi pesanan.

#### A. Arsitektur Sistem

Arsitektur sistem dapat digambarkan sebagai berikut.

##### 1. Tingkat 1: Backend API

*Backend API* menjadi lapisan utama yang menyediakan endpoint untuk menerima dan mengirim data antara aplikasi dan sistem pusat.

- Basis URL: <https://be.tifa-app.my.id/api/app/>
- Metode: HTTP POST dan GET digunakan untuk pertukaran data.
- Contoh *Endpoint*:
  - *POST /orders* – Menyimpan data pemesanan makanan.
  - *DELETE /coordinates/{table\_number}* – Menghapus koordinat dari nomor meja tertentu.
  - *GET /coordinates/{table\_number}* – Mengambil koordinat dari nomor meja.

##### 2. Tingkat 2: Database

Basis data menggunakan dua tabel utama:

- *koor*: Menyimpan pemetaan *table\_number* ke *coordinates*.
- *orders*: Menyimpan data pesanan seperti *order\_number*, *tray\_position*, *table\_number*, *coordinates*, serta *created\_at*.

##### 3. Tingkat 3: Repository Layer

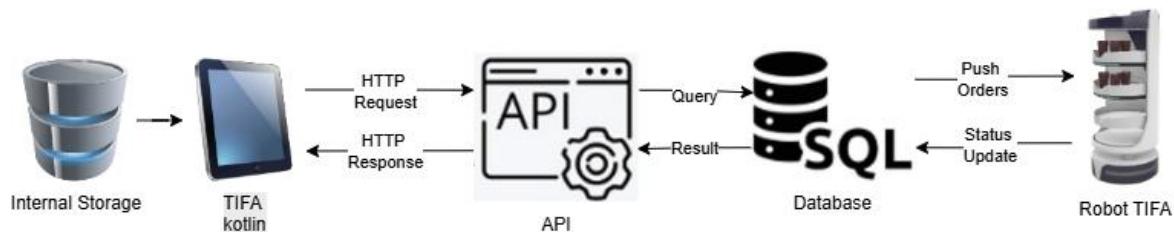
*AntarRepository.kt* mengatur komunikasi antara aplikasi dan *backend* menggunakan protokol HTTP berbasis JSON.

#### 4. Tingkat 4: User Interface (Android)

- Pengguna memilih posisi *tray* dan nomor meja melalui antarmuka Android yang dibangun dengan Jetpack Compose.
- Data yang dipilih akan ditangani oleh *AntarViewModel.kt* yang mengelola *state* dan alur data.

#### B. Alur Kerja Aplikasi

Alur kerja sistem dapat digambarkan sebagai berikut.



Gambar 4. 2 Alur Kerja Sistem Aplikasi

#### 1. Pengambilan Data Meja

Saat pengguna membuka mode antar pada aplikasi TIFA berbasis Kotlin, aplikasi akan mengirimkan HTTP Request ke API *endpoint* /coordinates untuk mendapatkan daftar meja beserta koordinatnya dari database. API kemudian memberikan HTTP Response yang berisi data tersebut dan aplikasi menampilkan daftar meja kepada pengguna.

#### 2. Pemilihan *Tray* dan Meja

Pengguna memilih kombinasi *tray* (Atas, Tengah, Bawah) dan meja (misalnya Meja 1, Meja 2, dan seterusnya). Setelah itu, aplikasi secara otomatis memanggil API get\_coordinates.php dengan parameter nomor meja yang dipilih untuk mengambil koordinat yang sesuai dari database.

#### 3. Validasi dan Konfirmasi

Data yang didapatkan dari pemilihan *tray* dan meja tersebut dikonversi menjadi objek *AntarData* dalam aplikasi dan disimpan sementara dalam daftar pesanan. Setelah semua pilihan dan data dipastikan benar, aplikasi mengirimkan data tersebut melalui API /orders untuk disimpan ke dalam tabel orders di database.

## 4. Pengambilan Data Terbaru

API /latest-orders digunakan untuk mengambil maksimal tiga data pesanan terbaru dari database. Data ini kemudian dikirimkan ke Robot TIFA /most-recent-orders sebagai perintah pengantar. Robot TIFA menerima data pesanan ini untuk menjalankan fungsi pengantaran, dan setelah selesai, status pesanan di-update kembali ke database menggunakan API yang sesuai.

### 4.2.2 Sub-Sistem *Backend* API

Sub-sistem *backend* API dalam mode *antar* pada aplikasi TIFA bertanggung jawab untuk menerima permintaan data dari aplikasi Android, memproses logika bisnis terkait pesanan antar, serta berinteraksi dengan basis data dbtes. *Backend* dibangun menggunakan skrip PHP yang masing-masing memiliki fungsi spesifik terhadap data pemesanan maupun koordinat meja.

#### A. /orders– Menyimpan Data Pesanan

File orders.php merupakan *endpoint* utama untuk menyimpan data pesanan pengguna ke dalam tabel orders.

```
// Ambil dan validasi input JSON dari request body
$input = $this->validateJsonInput();
// Cek input wajib: order_number dan tray_position harus diisi
if (empty($input['order_number']) || empty($input['tray_position'])) {
    $this->sendResponse(400, 'Kolom wajib diisi.');
    return;
}
// Siapkan data pesanan yang akan disimpan ke database
$dataToCreate = [
    'robot_id' => $input['robot_id'] ?? null,           // Opsional,
    bisa null
    'order_number' => $input['order_number'],           // Nomor
    pesanan (wajib)
    'tray_position' => $input['tray_position'],         // Posisi tray
    (wajib)
    'status' => $input['status'] ?? 0,                  // Status awal
    default = 0];
// Simpan data pesanan ke database lewat model
$newOrderId = $this->orderModel->create($dataToCreate);
// Jika gagal menyimpan (ID false), kirim error response
if ($newOrderId === false) {
    $this->sendResponse(500, 'Gagal menyimpan pesanan.');
    return;
}
// Ambil ulang data pesanan yang baru saja dibuat, berdasarkan ID
$createdOrder = $this->orderModel->findById((int) $newOrderId);
// Jika gagal mengambil ulang data, kirim error
if (!$createdOrder) {
    $this->sendResponse(500, 'Pesanan tersimpan tapi gagal
    diambil.');
    return;
}
```

```

// Kirim notifikasi log ke sistem lain melalui Socket.IO
$this->socketNotifier->notify('order_log_event_from_php', [
    'order_id' => (int) $createdOrder['id'], // ID
    'action_type' => 'created', // Aksi: pembuatan
    'message' => "Pesanan baru ID {$createdOrder['id']} ", // Pesan log];
    // Kirim response sukses (HTTP 201) dengan data pesanan
    $this->sendResponse(201, 'Pesanan berhasil dibuat.', [
        'order' => $createdOrder]);

```

File ini bekerja dengan skema sebagai berikut:

1. Validasi *Input*: Menerima data dalam format JSON dan memastikan `order_number` dan `tray_position` diisi.
2. Pengambilan Koordinat: Berdasarkan `table_number`, file mengambil koordinat dari tabel `koor` menggunakan *prepared statement* SQL.
3. Penyimpanan ke Tabel `orders`: Setelah koordinat ditemukan, data lengkap (`order_number`, `tray_position`, `table_number`, dan `coordinates`) disimpan ke tabel `orders`.
4. Kirim Notifikasi: Notifikasi log dikirim ke sistem eksternal via Socket.IO.
5. *Respons JSON*: Mengembalikan *respons* JSON berisi detail pesanan atau pesan *error*.



```

1  {
2      "status": "success",
3      "message": "Pesanan berhasil dibuat.",
4      "data": {
5          "order": {
6              "id": 30,
7              "robot_id": 3,
8              "order_number": 1,
9              "tray_position": "Atas",
10             "table_number": "Meja 1",
11             "coordinates": "[6.0,1.0]",
12             "status": 0,
13             "created_at": "2025-07-06 06:15:48"
14         }
15     }
16 }

```

**Gambar 4.3 Postman insert\_order.php**

Pada Postman menampilkan sebuah respon dalam format JSON yang menunjukkan bahwa sebuah proses berhasil dijalankan. Hal ini ditandai dengan nilai "success" yang bernilai *true*.

## B. /coordinates– Mendapatkan Koordinat Meja

*Endpoint* ini menerima `table_number` dan mengembalikan nilai `coordinates` dalam bentuk JSON. File ini berguna ketika aplikasi ingin mengetahui posisi koordinat meja sebelum menyimpan data pesanan.

```
// Cek apakah parameter table_number kosong
if (empty($tableNumber)) {
    $this->sendResponse(400, 'Parameter "table_number" wajib ada.', [
['success' => false]);
    return;
}
try {
    // Siapkan query untuk mengambil koordinat berdasarkan nomor meja
    $stmt = $this->pdo->prepare("SELECT coordinates FROM koor WHERE
table_number = :table_number");
    // Binding parameter dengan tipe string
    $stmt->bindParam(':table_number', $tableNumber, PDO::PARAM_STR);
    // Eksekusi query
    $stmt->execute();
    // Ambil hasil sebagai array asosiatif
    $row = $stmt->fetch(PDO::FETCH_ASSOC);
    if ($row) {
        // Jika data ditemukan, ambil nilai koordinat dan kirim
        response sukses
        $coordinatesValue = $row['coordinates'];
        $this->sendResponse(200, 'Koordinat berhasil ditemukan.', [
            'success' => true,
            'coordinates' => $coordinatesValue
        ]);
    } else {
        // Jika tidak ditemukan, kirim response 404
        $this->sendResponse(404, 'Meja tidak ditemukan.', ['success' => false]);
    }
} catch (PDOException $e) {
    // Tangani error database dan kirim response 500
    error_log("Database error: " . $e->getMessage());
    $this->sendResponse(500, 'Terjadi masalah database.', ['success' => false]);
} catch (Throwable $e) {
    // Tangani error tak terduga dan kirim response 500
    error_log("Unexpected error: " . $e->getMessage());
    $this->sendResponse(500, 'Terjadi kesalahan internal.', ['success' => false]);
}
```

File ini bekerja dengan skema sebagai berikut:

1. Pemeriksaan Parameter: Mengecek apakah parameter `table_number` dikirim melalui URL. Jika kosong, sistem mengembalikan *response* JSON dengan status 400 dan pesan bahwa parameter wajib ada.

2. *Query Database Aman*: Menggunakan *prepared statement* untuk mengambil nilai coordinates dari tabel koor berdasarkan table\_number. Parameter di-bind sebagai string (PDO::PARAM\_STR) untuk mencegah SQL *injection*.
3. Pengambilan dan Pengembalian Data: Jika data ditemukan, sistem mengembalikan coordinates dalam format JSON dengan status 200. Jika tidak ditemukan, sistem mengembalikan *responses* JSON dengan status 404 dan pesan "Meja tidak ditemukan".



```

1  {
2      "status": "success",
3      "message": "Data semua koordinat berhasil diambil.",
4      "tables": [
5          {
6              "id": 1,
7              "table_number": "Meja 1",
8              "coordinates": "6.0,1.0"
9          },
10         {
11             "id": 2,
12             "table_number": "Meja 2",
13             "coordinates": "12.1,6.0"
14         },
15         {
16             "id": 7,
17             "table_number": "Meja 3",
18             "coordinates": "3.4,5.6"
19         }
20     ]
21 }

```

**Gambar 4. 4 Postman get\_coordinates.php**

Output yang ditampilkan di Postman berupa objek JSON. Ini menunjukkan bahwa permintaan GET ke *endpoint* berhasil memproses parameter table\_number dan menemukan data yang sesuai di database.

### C. /latest-orders– Mengambil Data Pesanan Terbaru

File ini digunakan untuk mengambil daftar pesanan terbaru berdasarkan created\_at terbaru, terbatas hingga 3 entri. Tujuannya agar hanya pesanan aktif yang dikirim ke sistem antar (robot atau perangkat eksekusi).

```

try {// Ambil timestamp pesanan terbaru dari tabel orders
    $stmt = $this->pdo->prepare("SELECT created_at FROM orders ORDER
BY created_at DESC LIMIT 1");
    $stmt->execute();
    $latestTimestamp = $stmt->fetchColumn(); // hanya ambil kolom
pertama (created_at)

```

```

// Jika tidak ada data pesanan, kembalikan response kosong
if (!$_latestTimestamp) {
    $this->sendResponse(200, 'Tidak ada pesanan ditemukan.',
['orders' => []]);
    return;
// Hitung toleransi waktu 1 detik sebelum timestamp terbaru
$toleranceTime = date('Y-m-d H:i:s', strtotime($_latestTimestamp)
- 1);
// Ambil maksimal 3 pesanan yang memiliki timestamp dalam
rentang toleransi
$query = "
    SELECT order_number, tray_position, table_number, status
    FROM orders
    WHERE created_at BETWEEN :tolerance_time AND
:latest_timestamp
    ORDER BY created_at DESC, id DESC
    LIMIT 3";
$stmt = $this->pdo->prepare($query);
$stmt->bindParam(':tolerance_time', $toleranceTime);
$stmt->bindParam(':latest_timestamp', $_latestTimestamp);
$stmt->execute();
$data = $stmt->fetchAll(PDO::FETCH_ASSOC); // ambil hasil
sebagai array asosiatif
// Jika tidak ada data dalam toleransi, ambil 1 pesanan terakhir
saja
if (empty($data)) {
    $stmt = $this->pdo->prepare("
        SELECT order_number, tray_position, table_number, status
        FROM orders
        ORDER BY created_at DESC
        LIMIT 1");
    $stmt->execute();
    $singleRow = $stmt->fetch(PDO::FETCH_ASSOC);
    if ($singleRow) {
        $data[] = $singleRow;
    }
// Balik urutan data agar yang pertama dibuat muncul di atas
$data = array_reverse($data);
// Kirim response JSON berisi data pesanan terbaru
$this->sendResponse(200, 'Data pesanan terbaru berhasil
diambil.', ['orders' => $data]);
} catch (PDOException $e) {
    // Tangani kesalahan database
    error_log("Database error: " . $e->getMessage());
    $this->sendResponse(500, 'Terjadi masalah database.',
['error_detail' => $e->getMessage()]);
} catch (Throwable $e) {
    // Tangani kesalahan tak terduga
    error_log("Unexpected error: " . $e->getMessage());
    $this->sendResponse(500, 'Terjadi kesalahan internal.',
['error_detail' => $e->getMessage()]);
}

```

File ini bekerja dengan skema sebagai berikut:

1. Pengambilan Timestamp: *Query* pertama mengambil `created_at` paling akhir dari tabel `orders`, disimpan dalam `$_latestTimestamp`.

2. Query Data Berdasarkan *Timestamp*: Mengambil maksimal 3 pesanan dengan *created\_at* antara 1 detik sebelum hingga sama dengan \$latestTimestamp. Field yang diambil: *order\_number*, *tray\_position*, *table\_number*, dan *status*.
3. Kondisi Cadangan: Jika tidak ada data dalam rentang tersebut, sistem menjalankan *query* cadangan untuk mengambil 1 pesanan terakhir.

The screenshot shows the Postman interface with the following JSON response:

```

1  {
2      "status": "success",
3      "message": "Data pesanan terbaru berhasil diambil.",
4      "orders": [
5          {
6              "order_number": 1,
7              "tray_position": "Atas",
8              "table_number": "Meja 1",
9              "status": 0
10         }
11     ]
12 }

```

**Gambar 4. 5 Postman latest-orders.php**

Postman menampilkan JSON yang berisi informasi pesanan terakhir berdasarkan *created\_at* terbaru dalam database.

#### D. /most-recent-orders – Mengambil Data Pesanan Terkini

File deliver.php digunakan oleh sistem robot atau perangkat eksekusi untuk mengambil tiga data pesanan terakhir yang telah disimpan di tabel orders. File ini ditulis dalam PHP dan menghasilkan *respons* dalam format JSON.

```

try {// Ambil 3 pesanan terakhir berdasarkan waktu dibuat (created_at
terbaru)
    $stmt = $this->pdo->prepare("
        SELECT id, robot_id, order_number, tray_position,
table_number, coordinates, status, created_at
        FROM orders
        ORDER BY created_at DESC
        LIMIT 3");
    $stmt->execute();
    // Ambil hasil query sebagai array asosiatif
    $data = $stmt->fetchAll(PDO::FETCH_ASSOC);
    // Kirim data ke client dalam format JSON
    $this->sendResponse(200, 'Data pesanan paling baru berhasil
diambil.', ['orders' => $data]);
} catch (PDOException $e) {
    // Tangani error terkait database
    error_log("Database error: " . $e->getMessage());
}

```

```

        $this->sendResponse(500, 'Terjadi masalah database.', [
            'error_detail' => $e->getMessage()
        ]);
    } catch (Throwable $e) {
        // Tangani error tak terduga lainnya
        error_log("Unexpected error: " . $e->getMessage());
        $this->sendResponse(500, 'Terjadi kesalahan internal server.', [
            'error_detail' => $e->getMessage()
        ]);
    }
}

```

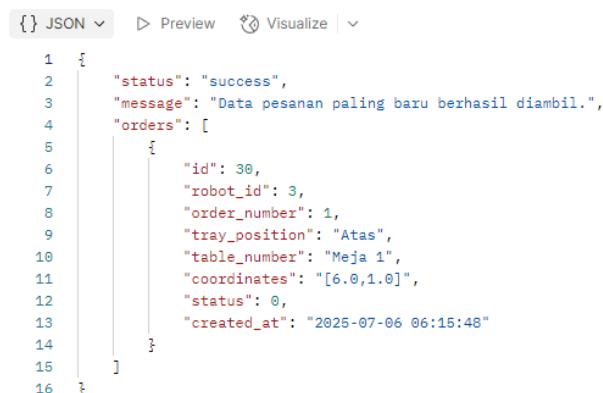
Data JSON yang dihasilkan dari most-recent-orders dikonsumsi oleh sistem robot pengantar yang akan:

1. Membaca order\_number untuk menentukan tray ke berapa yang diisi.
2. Mengambil coordinates sebagai parameter navigasi atau pemetaan rute.
3. Menyimpan atau menandai status pesanan menjadi selesai setelah pengantaran.

Sistem dapat mengakses *endpoint* ini secara berkala atau berdasarkan *trigger* dari tombol eksekusi. *Backend* juga menyediakan endpoint untuk menghapus atau menandai status pesanan, seperti:

1. /delete-order: Menghapus entri berdasarkan order\_number.
2. /update-order-status: Menandai pesanan sebagai selesai (status = 1).

Hal ini memastikan bahwa *backend* hanya menyimpan pesanan aktif dan menghindari duplikasi atau konflik eksekusi.



```

{
    "status": "success",
    "message": "Data pesanan paling baru berhasil diambil.",
    "orders": [
        {
            "id": 30,
            "robot_id": 3,
            "order_number": 1,
            "tray_position": "Atas",
            "table_number": "Meja 1",
            "coordinates": "[6.0,1.0]",
            "status": 0,
            "created_at": "2025-07-06 06:15:48"
        }
    ]
}

```

**Gambar 4. 6 Postman deliver.php**

Postman menampilkan dari tabel orders dalam format JSON. Setiap entri mencakup informasi lengkap pesanan seperti id, robot\_id, order\_number, posisi (tray\_position), nomor meja (table\_number), koordinat (coordinates), waktu dibuat (created\_at), dan status (status).

## E. Struktur Tabel Database

Basis data yang digunakan adalah MySQL dengan nama database dbtes. Berikut adalah struktur tabel yang relevan.

1. Tabel orders: untuk menyimpan pesanan yang dipilih oleh *user*.

	id	robot_id	order_number	tray_position	table_number	coordinates	created_at	status
▶	30	3	1	Atas	Meja 1	[6.0,1.0]	2025-07-06 06:15:48	0
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

**Gambar 4. 7 Tabel database orders**

- Kolom: id (INT, Primary Key, Auto Increment), robot\_id (INT): ID robot yang ditugaskan, order\_number (INT), tray\_position (VARCHAR), table\_number (VARCHAR), coordinates (VARCHAR), created\_at (TIMESTAMP), status (INT).
- Fungsi: Menyimpan detail pesanan pengiriman, termasuk nomor pesanan, posisi *tray*, nomor meja, koordinat, waktu pembuatan, dan status (0: pending, 1: selesai).
- 2. Tabel koor: untuk menyimpan koordinat yang disimpan oleh *user*.

	id	table_number	coordinates
▶	1	Meja 1	6.0,1.0
	2	Meja 2	12.1,6.0
	7	Meja 3	3.4,5.6
*	NULL	NULL	NULL

**Gambar 4. 8 Tabel database koor**

- Kolom: id (INT, Primary Key, Auto Increment), table\_number (VARCHAR), coordinates (VARCHAR).
- Fungsi: Memetakan nomor meja ke koordinat untuk keperluan navigasi. Contoh Data: ID 1, table\_number "Meja 1", coordinates 6.0,1.0.

## F. Keamanan dan *Respons*

Semua file *backend* dilengkapi dengan:

1. *Header Content-Type*: application/json
2. Mekanisme validasi *input* parameter
3. Penanganan kesalahan koneksi database

#### 4. Struktur *responses* JSON yang konsisten

##### 4.2.3 Sub-Sistem AntarRepository.kt

AntarRepository.kt merupakan lapisan repository pada arsitektur aplikasi TIFA yang berfungsi sebagai penghubung antara *ViewModel* dengan *backend server*. Sub-sistem ini menjalankan seluruh komunikasi data menggunakan protokol HTTP dan format pertukaran data JSON. Implementasi antar muka jaringan dilakukan menggunakan kelas HttpURLConnection bawaan Java serta coroutine dari Kotlin untuk operasi asinkronus.

###### A. Fungsi Utama Repository

Repository ini dirancang secara modular untuk menangani berbagai permintaan data yang berkaitan dengan mode antar, meliputi:

###### 1. getTables

```
suspend fun getTables(): List<Table> = withContext(ioDispatcher) {  
    fetchTablesFromDatabase()  
}
```

getTables(): Mengambil daftar meja beserta koordinat dari database melalui get\_tables.php.

###### 2. getTableCoordinates

```
suspend fun getTableCoordinates(tableName: String): String? =  
withContext(ioDispatcher) {  
    try { // Encode nama meja agar aman digunakan di URL  
        val encodedTableName = URLEncoder.encode(tableName, "UTF-8")  
        // Buat URL endpoint dengan parameter table_number  
        val url  
        =URL("$baseUrl/api/app/coordinates?table_number=$encodedTableName")  
        // Buka koneksi HTTP ke server  
        val connection = url.openConnection() as HttpURLConnection  
        connection.requestMethod = "GET" // Metode HTTP GET  
        connection.connectTimeout = 5000 // Timeout koneksi 5 detik  
        connection.readTimeout = 5000 // Timeout pembacaan 5 detik  
        if (connection.responseCode == 200) {  
            // Baca respons dari server  
        }  
    }  
}
```

```

        val response =
connection.inputStream.bufferedReader().use { it.readText() }
        // Ubah respons string menjadi JSON object
        val json = JSONObject(response)
        // Ambil array "tables" dari JSON
        val tables = json.optJSONArray("tables") ?:
return@withContext null
        // Loop semua objek meja dan cari yang sesuai
        for (i in 0 until tables.length()) {
            val obj = tables.getJSONObject(i)
            if (obj.getString("table_number") == tableName) {
                // Return koordinat jika meja ditemukan
                return@withContext
obj.getString("coordinates")}}
        null // Return null jika tidak ada meja yang cocok
    } else null // Return null jika respons bukan 200
} catch (e: Exception) {
    null // Return null jika terjadi error (misalnya koneksi gagal)}}

```

`getTableCoordinates(tableName)`: Mengambil nilai koordinat berdasarkan nama meja tertentu dari `/coordinates`.

### 3. sendOrderData

```

suspend fun sendOrderData(data: List<AntarData>): Result<Unit> =
withContext(ioDispatcher) {
    try {data.forEach { item ->
        // Buat URL endpoint untuk menyimpan pesanan
        val url = URL("$baseUrl/api/orders")
        val connection = url.openConnection() as
HttpURLConnection
        connection.requestMethod = "POST" // Gunakan metode POST
        connection.doOutput = true // Aktifkan pengiriman data
ke server
        connection.setRequestProperty("Content-Type",
"application/json") // Header tipe data JSON
        connection.connectTimeout = 5000 // Timeout koneksi 5
detik
        connection.readTimeout = 5000 // Timeout pembacaan 5
detik
        // Siapkan data JSON yang akan dikirim
        val jsonInput = JSONObject().apply {
            put("order_number", item.order) // Nomor pesanan
            put("tray_position", item.trayPosition) // Posisi
tray
            put("table_number", item.tableNumber) // Nama/meja
tujuan
            put("coordinates", item.coordinates) // Koordinat
meja
            put("robot_id", item.robotId) // ID robot yang
mengantar}
        // Kirim data JSON ke server
        connection.getOutputStream().write(jsonInput.toString().toByteArray())
    }
}

```

`sendOrderData(data: List<AntarData>): Mengirim seluruh data pesanan ke endpoint insert_order.php dalam bentuk JSON array.`

#### 4. `getMostRecentOrders`

```
suspend fun getMostRecentOrders(): List<OrderData> =  
withContext(ioDispatcher) {  
    try {// Buat URL ke endpoint backend yang menyediakan 3 pesanan  
    terbaru  
        val url = URL("$baseUrl/api/app/most-recent-orders")  
        // Buka koneksi HTTP  
        val connection = url.openConnection() as HttpURLConnection  
        connection.requestMethod = "GET" // Gunakan metode GET  
        connection.connectTimeout = 5000 // Timeout koneksi 5 detik  
        connection.readTimeout = 5000 // Timeout baca 5 detik  
        // Jika respons sukses (200 OK)  
        if (connection.responseCode == 200) {  
            // Baca respons dari input stream  
            val response =  
connection.inputStream.bufferedReader().use { it.readText() }  
            // Ubah string respons menjadi JSON  
            val json = JSONObject(response)  
            val jsonArray = json.getJSONArray("orders")  
            val orders = mutableListOf<OrderData>()  
            // Loop tiap elemen dan ubah ke dalam objek OrderData  
            for (i in 0 until jsonArray.length()) {  
                val obj = jsonArray.getJSONObject(i)  
                orders.add(  
                    OrderData(  
                        order_number = obj.getInt("order_number"),  
                        tray_position =  
obj.getString("tray_position"),  
                        table_number = obj.getString("table_number"),  
                        status = obj.getInt("status")))  
            }  
            orders // Kembalikan list pesanan  
        } else emptyList() // Jika bukan 200, kembalikan list kosong  
    } catch (e: Exception) {  
        emptyList() // Jika error, kembalikan list kosong}  
}
```

`getMostRecentOrders(): Mengambil 3 pesanan terbaru dari endpoint /most-recent-orders dan mengubahnya menjadi list OrderData.`

### B. Contoh Implementasi: Pengiriman Pesanan

```
val jsonInput = JSONObject().apply {  
    put("order_number", item.order)  
    put("tray_position", item.trayPosition)  
    put("table_number", item.tableNumber)  
    put("coordinates", item.coordinates) // from table `koor`  
    put("robot_id", item.robotId)}  
connection.outputStream.write(jsonInput.toString().toByteArray())
```

1. Fungsi ini akan melakukan iterasi untuk setiap data AntarData dan mengirimnya satu per satu.
2. Jika terjadi *error* selama pengiriman, maka proses akan dihentikan dan Result.failure akan dikembalikan.

#### **4.2.3.1 Penanganan Error dan Timeout**

Setiap koneksi ke *backend* dilengkapi dengan:

1. *Timeout* 5000 ms untuk koneksi dan pembacaan data.
2. *Try-catch exception handling* untuk memastikan kegagalan jaringan atau parsing JSON ditangani dengan baik.

#### **4.2.4 Sub-Sistem AntarViewModel.kt**

AntarViewModel.kt merupakan bagian dari arsitektur Model–View–View Model (MVVM) yang mengelola logika aplikasi dan status antarmuka pengguna untuk fitur mode antar. Komponen ini berperan sebagai jembatan antara antarmuka pengguna (Jetpack Compose) dengan lapisan repository (AntarRepository.kt), serta bertanggung jawab untuk memproses *input* pengguna, validasi urutan pesanan, dan mengatur status tampilan seperti loading, *error*, dan dialog konfirmasi.

##### **A. Struktur dan Variabel Status**

File ini menggunakan *StateFlow* dan *mutable state off* untuk memantau perubahan data dan memicu rekomposisi UI. Beberapa variabel penting antara lain:

1. selectedTray – Menyimpan *tray* yang dipilih pengguna

```
private var _selectedTray by mutableStateOf<String?>(null)
fun setSelectedTray(tray: String?) {
    _selectedTray = tray
    _selectedTable = null}
```

- *\_selectedTray* menyimpan pilihan *tray* (Atas, Tengah, atau Bawah) dalam *state*.
- *selectedTray* adalah akses publik untuk membaca nilai *tray* yang dipilih.
- Fungsi *setSelectedTray()* digunakan untuk mengatur *tray* yang dipilih, dan otomatis me-reset *selectedTable* agar tidak tertukar antar-*tray*.

## 2. selectedTable – Menyimpan meja yang dipilih pengguna

```
private var _selectedTable by mutableStateOf<String?>(null)
val selectedTable: String? get() = _selectedTable
fun setSelectedTable(table: String?) {
    _selectedTable = table}
```

- `_selectedTable` menyimpan nama meja yang dipilih.
- Getter `selectedTable` memungkinkan akses dari luar.
- Fungsi `setSelectedTable()` mengatur meja yang dipilih berdasarkan *input* pengguna.

## 3. selectedItems – Daftar objek pesanan

```
val selectedItems = mutableStateListOf<AntarData>()
```

- `selectedItems` menyimpan daftar data pesanan dalam bentuk objek `AntarData`.
- Data ini mencakup informasi *tray*, meja, dan koordinat.
- Digunakan untuk validasi urutan, pengiriman data, dan manipulasi (add/remove).

## 4. isLoading – Penanda status proses asinkronus

```
var isLoading by mutableStateOf(false)
```

- Dipakai untuk menandai apakah aplikasi sedang memproses data, seperti saat memuat atau mengirim ke server.
- Nilai true akan memicu UI menampilkan indikator loading.

## 5. showAddTableDialog, showConfirmationDialog – Penanda status dialog aktif

```
var showAddTableDialog by mutableStateOf(false)
var showConfirmationDialog by mutableStateOf(false)
fun showAddTableDialog() {
    showAddTableDialog = true}
result.onSuccess {
    selectedItems.clear()
    showConfirmationDialog = false
}.onFailure { e ->
    errorMessage = e.message ?: "Gagal mengirim data"}
```

- `showConfirmationDialog` untuk dialog konfirmasi saat *user* ingin mengirim data.
- Nilai true berarti dialog akan tampil di UI

## 6. *error* Message – Menyimpan pesan kesalahan

```
var errorMessage by mutableStateOf<String?>(null)
fun dismissError() {
    errorMessage = null}
```

- Menyimpan pesan kesalahan saat terjadi *error* (misal saat koneksi, validasi, atau proses gagal).
- Bisa ditampilkan ke pengguna melalui UI.
- *dismissError()* dipanggil untuk menghapus pesan setelah ditampilkan.

## B. Fungsi Utama ViewModel

Beberapa fungsi utama yang menangani logika mode antar:

1. *addSelection()*

Menambahkan pasangan *tray*–meja ke dalam *selectedItems*. Fungsi ini akan:

- Memastikan jumlah maksimal pesanan adalah 3.
- Mengambil koordinat dari server berdasarkan *table\_number*.
- Mengatur ulang pilihan setelah berhasil ditambahkan.

2. *confirmSelection()*

Fungsi ini akan:

- Melakukan validasi urutan pesanan agar berurutan (1, 2, 3).
- Mengirim data ke server menggunakan *AntarRepository.sendOrderData()*.
- Menghapus seluruh daftar setelah pengiriman berhasil.

3. *getOrderForTable(tableName)*: Mengambil urutan pesanan dari nama meja tertentu.

4. *removeSelection(trayPosition)*: Menghapus pesanan berdasarkan posisi *tray*.

5. *addTable()* dan *deleteTable()*: Menambahkan atau menghapus meja dari basis data koor.

6. *loadTables()*: Mengambil daftar meja dan memperbarui *StateFlow*.

## C. Validasi Urutan Pesanan

Salah satu aspek penting adalah validasi urutan:

```

val orders = selectedItems.map { it.order }.sorted()
val expected = (1..selectedItems.size).toList()
if (orders != expected) {
    throw IllegalArgumentException("Urutan harus berkelanjutan
1, 2, 3") }
```

Validasi ini mencegah pengguna mengirim data acak dan memastikan sistem robot dapat mengeksekusi sesuai urutan *tray*.

#### 4.2.4.1 Pengelolaan Error

*Error* ditangani dengan baik melalui:

1. *Try-catch blok* untuk fungsi asynchronous.
2. Penyimpanan pesan *error* ke *error Message* yang dapat langsung digunakan untuk notifikasi UI.

#### 4.2.5 Sub-Sistem Struktur Data: AntarData.kt dan OrderData.kt

Sub-sistem ini berperan dalam mendefinisikan struktur data yang digunakan untuk merepresentasikan informasi penting pada proses antar pesanan. File AntarData.kt dan OrderData.kt merupakan bagian dari lapisan *model* dalam arsitektur MVVM aplikasi Android. Struktur data ini digunakan sebagai objek transfer antara antarmuka pengguna (ViewModel), repository, hingga *backend* server melalui format JSON. Definisi data yang konsisten penting untuk memastikan integritas data selama proses komunikasi antar lapisan.

##### A. AntarData.kt – Model Data Antar

```
data class AntarData(  
    val order: Int,  
    val trayPosition: String,  
    val tableNumber: String,  
    val coordinates: String = "",  
    val robotId: Int = 3)
```

1. *order* (Integer): Menunjukkan nomor urutan pesanan, misalnya 1, 2, atau 3, yang merepresentasikan urutan *tray* pada robot pengantar.
2. *trayPosition* (String): Menyatakan posisi *tray* yang digunakan, dengan nilai yang mungkin yaitu "Atas", "Tengah", atau "Bawah".
3. *tableNumber* (String): Nama atau ID meja tujuan (contoh: "Meja 1").
4. *coordinates* (String): Koordinat meja yang diambil dari tabel koor di *backend*.
5. *robotId* (Int): ID robot yang mengantar (default: 3).

Penggunaan:

Objek ini dikonversi menjadi JSON dan dikirim satu per satu ke *endpoint /orders*.

##### B. Table – Model Data Meja

Kelas ini juga berada di dalam AntarData.kt dan digunakan untuk menyimpan daftar meja yang diambil dari server.

```
data class Table(
    val id: Int,
    val name: String,
    val coordinates: String,
    val x: Double? = null,
    val y: Double? = null)
```

1. id (Integer): ID unik yang digunakan untuk mengidentifikasi setiap entri meja di dalam database.
2. name (String): Nama meja yang digunakan secara deskriptif, seperti "Meja 1", "Meja 2", dan sebagainya.
3. coordinates (String): Koordinat meja sebagai referensi posisi dalam sistem navigasi robot.
4. x & y (Double): koordinat numerik untuk visualisasi peta atau keperluan lain.

Penggunaan:

Digunakan dalam proses pengambilan dan penambahan meja.

### C. OrderData.kt – Model Pesanan yang Tersimpan

Kelas ini digunakan untuk menyimpan data pesanan yang telah dikirim dan disimpan ke database. Data ini biasanya ditampilkan sebagai riwayat atau dikirim ke robot pengantar.

```
data class OrderData(
    val order_number: Int,
    val tray_position: String,
    val table_number: String,
    val status: Int)
```

1. order\_number (Integer): Nomor urutan pesanan yang menunjukkan posisi *tray*.
2. *tray\_position* (String): Posisi *tray*, seperti "Atas", "Tengah", atau "Bawah".
3. *table\_number* (String): ID atau nama meja tujuan pengantaran.
4. status (Integer): Status eksekusi pengantaran, dengan nilai:
  - 0: Belum dikirim
  - 1: Sudah selesai

Penggunaan:

Digunakan dalam fungsi getLatestOrders() di AntarRepository.kt untuk menampilkan status terkini pesanan di UI operator robot.

#### 4.2.6 Sub-Sistem Tampilan UI Mode Antar

Tampilan antarmuka pengguna (*User Interface/UI*) dari mode antar pada aplikasi TIFA dikembangkan menggunakan Jetpack Compose, *Framework* deklaratif modern untuk Android.

Fitur ini dirancang agar operator dapat dengan mudah memilih *tray* robot, menentukan meja tujuan, serta mengonfirmasi dan mengelola data pesanan dengan intuitif dan interaktif.

### A. Navigasi dari HomeScreen.kt

Pada halaman utama aplikasi (HomeScreen.kt), terdapat tombol navigasi dengan ikon Icons.Filled.RoomService dan label "Antar".

```
NavigationButton(
    icon = Icons.Filled.RoomService,
    text = "Antar",
    route = "antar",
    navController = navController)
```

Tombol ini berfungsi sebagai pemicu navigasi ke mode antar dengan route "antar", yang akan menampilkan halaman AntarScreen.

### B. Halaman Mode Antar — AntarScreen.kt

Halaman utama UI mode antar adalah AntarScreen, yang memuat elemen utama seperti pemilihan *tray*, pemilihan meja, daftar pesanan, tombol tambah dan konfirmasi, serta dialog-dialog pendukung.

```
@Composable
fun AntarScreen(navController: NavController) {
    val viewModel: AntarViewModel = viewModel(key = "antarKey")
    val tables by viewModel.tables.collectAsState(initial =
emptyList())
    val batteryLevel by viewModel.batteryLevel.collectAsState(initial
= 0)// Struktur UI utama menggunakan Scaffold
    Scaffold(topBar = { BatteryIndicator(batteryLevel = batteryLevel)
}, // Tampilkan indikator baterai di atas
        bottomBar = { BottomBar(navController) })
// Navigasi bawah) { padding ->
    Box(modifier = Modifier.padding(padding)) {
        BackgroundWithImage() // Latar belakang dengan gambar
        MainContent(viewModel, tables) // Konten utama milih meja
        LoadingIndicator(viewModel.isLoading) // Tampilkan
loading jika sedang proses
        ErrorDialog(viewModel.errorMessage) {
            viewModel.dismissError() } // Dialog error
        ConfirmationDialog(viewModel, navController) //
Konfirmasi pengiriman
        AddTableDialog(viewModel) // Dialog tambah meja}}}
```

AntarScreen menggunakan AntarViewModel untuk mengelola state aplikasi seperti daftar meja, level baterai, status loading, dan *error*. Scaffold utama pada AntarScreen terdiri dari:

1. BatteryIndicator di top bar untuk menampilkan status baterai.
2. BottomBar sebagai navigasi bawah.

3. Konten utama yang berisi komponen UI lain dan berbagai dialog.

### C. Elemen UI Utama — MainContent

MainContent mengatur *layout* utama yang terdiri dari:

```
@Composable
private fun MainContent(viewModel: AntarViewModel, tables:
List<Table>) {
    Column(
        modifier = Modifier.fillMaxSize().padding(16.dp),
        verticalArrangement = Arrangement.spacedBy(24.dp)
    ) { DoorControlSection(viewModel)
        TraySelectionSection(viewModel)
        TableGridSection(viewModel, tables)
        ActionButtonsSection(viewModel)
    }
}
```

Fungsi ini mengorganisasi seluruh UI utama mode antar secara terstruktur dan rapi menggunakan Jetpack Compose dengan memanfaatkan *state* dan logika dari AntarViewModel.

```
@Composable
private fun DoorControlSection(viewModel: AntarViewModel) {
    // Kartu berisi tombol kontrol pintu
    Card(
        shape = RoundedCornerShape(16.dp),
        colors = CardDefaults.cardColors(containerColor = Brown),
        modifier = Modifier.fillMaxWidth() {
            Column(modifier = Modifier
                .fillMaxWidth()
                .padding(16.dp),
                verticalArrangement = Arrangement.spacedBy(8.dp) ) {
                // Judul kontrol
                Text("Kontrol Pintu",
                    style = MaterialTheme.typography.titleLarge,
                    color = Color.White)
                // Tombol buka/tutup pintu
                Row(modifier = Modifier
                    .fillMaxWidth()
                    .padding(top = 8.dp),
                    horizontalArrangement = Arrangement.spacedBy(16.dp))
                {
                    // Tombol buka pintu
                    Button(onClick = { viewModel.sendDoorCommand(1) },
                        // Kirim perintah buka

```

```

        colors = ButtonDefaults.buttonColors(containerColor
= Color(0xFFFF8EBD9)),
                shape = RoundedCornerShape(8.dp),
                modifier = Modifier
                    .weight(1f)
                    .height(48.dp)) {
            Text("Buka Pintu", color = Brown, fontSize =
22.sp)
        }
        // Tombol tutup pintu
        Button(onClick = { viewModel.sendDoorCommand(0) }, //
Kirim perintah tutup
        colors =
ButtonDefaults.buttonColors(containerColor = Color(0xFFFF8EBD9)),
                shape = RoundedCornerShape(8.dp),
                modifier = Modifier
                    .weight(1f)
                    .height(48.dp) ) {
            Text("Tutup Pintu", color = Brown, fontSize =
22.sp) } } }
    }
}

```

UI ini memberikan kontrol sederhana dan langsung kepada pengguna untuk membuka atau menutup pintu robot pengantar. Fungsi tombol:

1. onClick tombol “Buka Pintu” memanggil viewModel.sendDoorCommand(1) — mengirim perintah buka.
2. onClick tombol “Tutup Pintu” memanggil viewModel.sendDoorCommand(0) — mengirim perintah tutup.

```

@Composable
private fun TraySelectionSection(viewModel: AntarViewModel) {
    // Kartu berisi pilihan tray (Atas, Tengah, Bawah)
    Card(shape = RoundedCornerShape(16.dp),
        colors = CardDefaults.cardColors(containerColor = Brown),
        modifier = Modifier.fillMaxWidth()) {
        Column(modifier = Modifier.padding(16.dp),
            verticalArrangement = Arrangement.spacedBy(8.dp)) {
            // Judul bagian
            Text("Pilih Tray", style =
MaterialTheme.typography.titleLarge, color = Color.White)
            // Deretan tombol pilihan tray
            Row(modifier = Modifier.fillMaxWidth(),
                horizontalArrangement = Arrangement.spacedBy(12.dp))
        } {
            listOf("Atas", "Tengah", "Bawah").forEach { tray ->
                Box(modifier = Modifier.weight(1f)) {
                    // Tombol tray, menandai yang sedang dipilih
                    TrayButton(
                        tray = tray,
                        isSelected = viewModel.selectedTray ==
tray, // Cek apakah tray sedang dipilih

```

```

        onSelect = {
    viewModel.setSelectedTray(tray) }) // Set tray yang dipilih {}
}
}
}
}

```

UI ini memudahkan pengguna memilih salah satu dari tiga posisi *tray* yang tersedia pada robot. Pemilihan *tray* bersifat tunggal (hanya satu yang dapat dipilih dalam satu waktu).

1. Properti `isSelected` menandakan apakah *tray* tersebut sedang dipilih (mengacu ke *state* `viewModel.selectedtray`).
2. Ketika tombol diklik, memanggil `viewModel.setSelectedtray` untuk mengupdate pilihan *tray* di ViewModel.

```

@Composable
private fun TableGridSection(viewModel: AntarViewModel, tables:
List<Table>) {
    // Hitung jumlah kolom berdasarkan lebar layar
    val columns = when {
        LocalConfiguration.current.screenWidthDp < 600 -> 4
        LocalConfiguration.current.screenWidthDp < 840 -> 6
        else -> 8
    }
    var showDelete by remember { mutableStateOf(false) }
    var selected by remember { mutableStateOf<String?>(null) }

    // Kartu berisi tombol aksi dan grid meja
    Card(shape = RoundedCornerShape(16.dp), colors =
CardDefaults.cardColors(Brown)) {
        Column(Modifier.padding(8.dp)) {
            // Header: judul dan tombol tambah/hapus
            Row(Modifier.fillMaxWidth(), Arrangement.SpaceBetween)
            {Text("Pilih Meja", color = Color.White)
                Row {Button(onClick = { viewModel.showAddTableDialog
= true }) { Text("Tambah", fontSize = 12.sp) }
                    Button(onClick = { showDelete = true }, colors =
ButtonDefaults.buttonColors(Color.Red)) {
                        Text("Hapus", fontSize = 12.sp)}}
            }
            // Grid meja
            LazyVerticalGrid(columns = GridCells.Fixed(columns)) {
                items(tables) {
                    TableButton(
                        table = it,
                        isSelected = viewModel.selectedTable ==
it.name,
                        onSelect = {
                            viewModel.setSelectedTable(it.name) },
                            viewModel = viewModel,
                            onClickForDelete = { selected = it.name }))}
            }
        }
    }
}

```

TableGridSection menyesuaikan grid meja secara *responsif* (4, 6, atau 8 kolom berdasarkan lebar layar) menggunakan BoxWithConstraints. UI dalam *Card* berlatar cokelat menampilkan judul, tombol "Tambah Meja" dan "Hapus Meja", serta LazyVerticalGrid untuk TableButton (maks 500dp). *State* showDeleteDialog dan tableToDelete kelola dialog hapus, dengan AntarViewModel untuk data meja.

1. *Responsif*: Kolom grid (4, 6, atau 8) sesuai lebar layar.
2. *State*: showDeleteDialog untuk dialog, tableToDelete simpan meja hapus.
3. UI: *Card*, *Column*, *Row* untuk judul, tombol; LazyVerticalGrid untuk meja.
4. ViewModel: Kelola pemilihan, tambah, hapus meja.
5. Fungsi: Grid *responsif*, tombol tambah/hapus, pilih meja

```
// Tampilkan dialog konfirmasi jika flag aktif
if (showDeleteDialog) {
    AlertDialog(
        onDismissRequest = { showDeleteDialog = false }, // Tutup
        dialog saat diklik di luar
        title = { Text("Hapus Meja") }, // Judul dialog
        // Konten utama dialog: daftar meja
        text = { Column(
            modifier = Modifier
                .fillMaxWidth()
                .heightIn(max = 200.dp)
                .verticalScroll(rememberScrollState()) // Biar
                konten bisa scroll) { Text("Pilih meja yang ingin dihapus:")
                    Spacer(modifier = Modifier.height(8.dp))
                    // Daftar meja dalam bentuk radio button
                    tables.forEach { table ->
                        Row(verticalAlignment =
                            Alignment.CenterVertically,
                            modifier = Modifier
                                .fillMaxWidth()
                                .padding(vertical = 4.dp)) {
                            RadioButton(
                                selected = tableToDelete == table.name,
                                // Cek jika meja ini terpilih
                                onClick = { tableToDelete = table.name }
                                // Simpan meja yang dipilih)
                            Text(text = table.name,

```

```

        modifier = Modifier.padding(start =
8.dp) ) } } } ,
    // Tombol "Hapus"
confirmButton = {
    Button(onClick = {
        tableToDelete?.let { name ->
            viewModel.deleteTable(name) // Hapus meja
via ViewModel
            tableToDelete = null}
            showDeleteDialog = false // Tutup dialog},
        enabled = tableToDelete != null, // Aktif jika meja
dipilih
            colors =ButtonDefaults.buttonColors(containerColor =
Color.Red) {Text("Hapus") } },
    // Tombol "Batal"
dismissButton = {Button(onClick = {
        showDeleteDialog = false // Tutup dialog
        tableToDelete = null // Reset pilihan}
    ) {Text("Batal") }}} }

```

Blok ini mengelola dialog hapus meja saat `showDeleteDialog` bernilai `true`, menggunakan `AlertDialog` dengan judul "Hapus Meja", daftar meja dalam `Column` (maks 200dp, scrollable), dan tombol "Hapus" serta "Batal". `RadioButton` dan `Text` menampilkan meja untuk dipilih (`tableToDelete`). Tombol "Hapus" aktif jika meja dipilih, memanggil `viewModel.deleteTable`, lalu menutup dialog.

1. Struktur Dialog: `AlertDialog` dengan judul, daftar meja, tombol "Hapus"/"Batal".
2. Isi Dialog: `Column` (maks 200dp, scroll), `RadioButton+Text` untuk pilih meja, `TextButton "Batal"`.
3. Tombol Aksi: "Hapus" (aktif jika `tableToDelete` ada, panggil `viewModel.deleteTable`), "Batal" tutup dialog.
4. ViewModel: Hapus meja via `viewModel.deleteTable`

## D. Dialog Konfirmasi Pesanan

```
@Composable
fun ConfirmationDialog(viewModel: AntarViewModel, navController: NavController) {
    if (viewModel.showConfirmationDialog) { // Tampilkan dialog jika flag aktif
        AlertDialog(
            onDismissRequest = { viewModel.showConfirmationDialog = false }, // Tutup dialog
            modifier = Modifier.fillMaxWidth(0.8f).padding(24.dp),
            title = { Text("Konfirmasi ${viewModel.selectedItems.size} Pesanan") }, // Judul dialog
            text = { // Daftar pesanan
                Column(modifier =
                    Modifier.fillMaxWidth().verticalScroll(rememberScrollState())) {
                    viewModel.selectedItems.sortedBy { it.order }.forEach { item ->
                        Row(
                            modifier =
                                Modifier.fillMaxWidth().padding(vertical = 4.dp),
                            verticalAlignment =
                                Alignment.CenterVertically,
                            horizontalArrangement =
                                Arrangement.SpaceBetween
                        ) {
                            Column { // Info pesanan
                                Text("${item.order}. ${item.trayPosition} → ${item.tableNumber}")
                                Text("Koordinat: ${item.coordinates}", color = Color.Gray)
                            }
                            IconButton(onClick = {
                                viewModel.removeSelection(item.trayPosition) // Hapus item
                                Icon(Icons.Default.Delete,
                                    contentDescription = "Hapus", tint = MaterialTheme.colorScheme.error)
                            })
                        }
                    }
                }
            },
            confirmButton = { // Tombol konfirmasi
                Button(
                    onClick = { viewModel.confirmSelection();
                    navController.navigate("loading") },
                    modifier = Modifier.fillMaxWidth(),
                    shape = RoundedCornerShape(20.dp),
                    colors =
                        ButtonDefaults.buttonColors(containerColor = ForestGreen)
                ) { Text("Konfirmasi") }
            },
            dismissButton = { // Tombol batal
                Button(
                    onClick = { viewModel.showConfirmationDialog = false },
                    modifier = Modifier.fillMaxWidth(),
                    shape = RoundedCornerShape(20.dp),
                    colors =
                        ButtonDefaults.buttonColors(containerColor = Color.Red)
                ) { Text("Batal") } })
    }
}
```

ConfirmationDialog menampilkan dialog saat viewModel.showConfirmationDialog aktif. Menampilkan daftar selectedItems yang bisa dihapus, lalu tombol Konfirmasi (simpan & navigasi ke "loading") dan Kembali Edit (tutup dialog).

## E. Mengambil Data Pesanan dan Menangani Pesanan Baru

```
LaunchedEffect(Unit) {
    while (true) {
        val latestOrders = repository.getLatestOrders()
        orderList = latestOrders
        val nextDelivered = latestOrders
            .filter { it.status == 1 }
            .firstOrNull { it.order_number != currentDeliveredOrder?.order_number }

        if (nextDelivered != null) {
            currentDeliveredOrder = nextDelivered}
        delay(3000)
    }
}
```

1. Polling *backend* setiap 3 detik untuk memperbarui daftar pesanan.
2. Memantau pesanan yang statusnya sudah terkirim (status == 1).
3. Jika ada pesanan terkirim baru, dialog konfirmasi muncul

## F. Daftar Tujuan Pengantaran

```
Card(
    colors = CardDefaults.cardColors(containerColor = Brown),
    shape = RoundedCornerShape(20.dp),
    modifier = Modifier.fillMaxWidth().weight(0.8f)
) {
    Column(modifier = Modifier.padding(16.dp)) {
        Text("Daftar Tujuan Pengantaran:", color = Color.White,
style = MaterialTheme.typography.titleLarge)
        Spacer(Modifier.height(12.dp))
        LazyColumn(verticalArrangement =
Arrangement.spacedBy(16.dp)) {
            items(orderList) { order ->
                Column(
                    modifier = Modifier.fillMaxWidth()
                        .background(Color.White.copy(alpha =
0.08f), shape = RoundedCornerShape(12.dp))
                        .padding(12.dp)
                ) {
                    val mejaOnly = order.table_number.filter {
it.isDigit() }
                    Text(text = "MEJA $mejaOnly", style =
MaterialTheme.typography.headlineLarge, color = Color.White)
                    Text(text = "Tray: ${order.tray_position}",
style = MaterialTheme.typography.titleMedium, color = Color.LightGray)
                }
            }
        }
    }
}
```

1. Menampilkan daftar pesanan aktif yang sedang diantar, dengan informasi nomor meja dan posisi *tray*.
2. Daftar dalam bentuk LazyColumn dengan *styling* kartu.

## G. Dialog Konfirmasi Pesanan Sampai

```
@Composable
fun KonfirmasiDialog(
    currentDeliveredOrder: OrderData,
    onKonfirmasi: () -> Unit,
    talkBackViewModel: TalkBackViewModel) {
    // Memicu suara konfirmasi saat dialog muncul
    LaunchedEffect(currentDeliveredOrder) {
        talkBackViewModel.speakConfirmationDialog(
            tableNumber = currentDeliveredOrder.table_number,
            trayPosition = currentDeliveredOrder.tray_position)
    }
    Dialog(onDismissRequest = {}) {
        Card(
            modifier = Modifier
                .fillMaxWidth()
                .padding(horizontal = 24.dp),
            shape = RoundedCornerShape(16.dp),
            colors = CardDefaults.cardColors(containerColor =
Color.White)) {
            Column(
                modifier = Modifier.padding(20.dp),
                horizontalAlignment =
Alignment.CenterHorizontally,
                verticalArrangement = Arrangement.spacedBy(16.dp))
            {
                Row(modifier = Modifier.fillMaxWidth(),
                    horizontalArrangement =
Arrangement.spacedBy(16.dp),
                    verticalAlignment =
Alignment.CenterVertically) {
                    //Animasi lottie melambai
                    val wavingComposition by
rememberLottieComposition(LottieCompositionSpec.RawRes(R.raw.waving))
                    val wavingProgress by
animateLottieCompositionAsState(
                        composition = wavingComposition,
                        iterations =
LottieConstants.IterateForever)
                    LottieAnimation(composition =
wavingComposition,
                        progress = { wavingProgress },
                        modifier = Modifier
                            .size(100.dp)
                            .weight(1f))
                    // Tombol konfirmasi ucapan terima kasih dan
aksi
                    Button(onClick = {
                        talkBackViewModel.speakThankYou()
                        onKonfirmasi()
                    }, /* properti lainnya */) {
                        Text(text = "Konfirmasi", color =
Color.White)
                    }
                }
            }
        }
    }
}
```

1. Muncul otomatis ketika ada pesanan dengan status terkirim.
2. Memanfaatkan Text-To-Speech (TTS) untuk konfirmasi suara.
3. Menampilkan animasi dan Detail pesanan.
4. Tombol konfirmasi untuk menghapus pesanan dari *backend* dan menutup dialog.

#### 4.2.7 Hasil Implementasi

Pada aplikasi *mobile* TIFA, halaman utama menampilkan daftar pesanan yang tersedia di Tel-U Coffee. Saat pertama kali dibuka, aplikasi akan menampilkan indikator *loading* untuk menunggu data pesanan diambil dari server melalui *AntarViewModel*. Setelah data berhasil dimuat, setiap pesanan ditampilkan dalam kartu terpisah dengan informasi ringkas berupa nomor pesanan, posisi *tray*, nomor meja, dan status pesanan.

Tiap kartu dirancang *responsif* menggunakan Jetpack Compose, memanfaatkan ElevatedCard, *Column*, dan *Row* untuk menyusun elemen-elemen UI secara vertikal dan horizontal. State expanded mengontrol tampilan detail pesanan yang dapat diperluas, sementara variabel status menentukan warna label status ("Dalam Proses" atau "Selesai").

Setelah data pesanan dimuat, kartu pesanan menampilkan label status berwarna hijau untuk "Dalam Proses" dan abu-abu untuk "Selesai". Judul kartu menampilkan nomor pesanan dalam teks tebal, diikuti oleh informasi posisi *tray* (misalnya, "Atas", "Tengah", "Bawah") dan nomor meja (misalnya, "Meja 5"). Detail pesanan dibatasi pada dua baris awal, dengan tombol "Tutup" untuk menyembunyikan atau memperluas teks sesuai nilai state expanded.

Ada dua tombol aksi disediakan: Kirim Perintah untuk mengirimkan perintah khusus ke robot, dan Konfirmasi Selesai yang hanya aktif jika status pesanan masih "Dalam Proses", memicu fungsi konfirmasi yang diimplementasikan di *ViewModel*.

#### 4.2.8 Penggunaan *Hardware* untuk Aplikasi

Untuk sisi aplikasi, digunakan Redmi Pad SE sebagai perangkat antarmuka pengontrol dan monitoring.

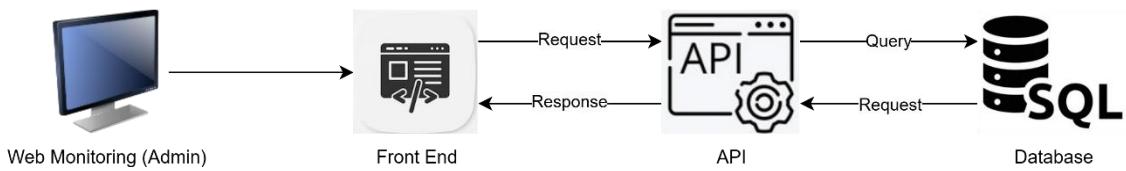
**Tabel 4. 1 Tabel Penggunaan *Hardware* Aplikasi**

Komponen	Spesifikasi
Tablet Aplikasi	Redmi Pad SE
Prosesor Tablet	Snapdragon® 680 Mobile Platform, 6nm, CPU 8 core hingga 2,4 GHz
RAM Tablet	LPDDR4X 4GB

Komponen	Spesifikasi
Penyimpanan Tablet	eMMC 5.1 128GB, mendukung ekspansi hingga 1TB
Layar Tablet	11 inci FHD+ (1920 x 1200), refresh rate hingga 90 Hz, kecerahan 400 nit
Konektivitas Tablet	Wi-Fi dual-band 2,4 GHz & 5 GHz, Bluetooth 5.0
Baterai Tablet	Kapasitas 8.000 mAh, pengisian daya 10W USB Type-C
Sistem Operasi Tablet	MIUI Pad 14 berbasis Android 13

#### 4.2.9 Detil Implementasi Web Monitoring

Skema web yang ditampilkan pada gambar 4.9 menggambarkan arsitektur dasar sistem aplikasi web yang terdiri dari empat komponen utama, yaitu web monitoring (admin), *frontend*, API, dan database. Web monitoring berfungsi sebagai antarmuka bagi administrator untuk mengakses dan mengelola sistem melalui *browser* atau perangkat client. *Frontend* merupakan bagian dari aplikasi yang bertugas menampilkan tampilan kepada pengguna. API bertindak sebagai perantara yang menghubungkan *frontend* dengan database, menjalankan logika, serta menangani komunikasi data. Database menggunakan SQL sebagai sistem manajemen data terstruktur yang menyimpan seluruh informasi aplikasi.



**Gambar 4. 9 Skema Web Monitoring**

#### A. Arsitektur Sistem

Website TIFA diimplementasikan menggunakan Vue.js sebagai *Frontend* dengan manajemen *state* menggunakan composable auth.js yang berfungsi untuk mengelola otentikasi pengguna melalui komunikasi API dengan *backend* PHP yang berjalan di server XAMPP. *Backend* PHP mengelola logika bisnis seperti validasi *user*, manajemen session, dan penyajian data dalam form JSON, serta berinteraksi dengan database MySQL untuk menyimpan data pengguna dan status baterai. Arsitektur sistem dapat digambarkan sebagai berikut.

##### 1. Data Layer (Database)

Lapisan data merupakan fondasi utama dari sistem TIFA yang berfungsi sebagai tempat penyimpanan seluruh informasi yang diperlukan dalam operasional sistem. Teknologi yang digunakan pada layer ini adalah MySQL, di mana beberapa tabel utama digunakan untuk menyimpan berbagai jenis data, antara lain: *battery\_data* untuk menyimpan informasi status dan performa baterai robot; *orders* untuk mencatat data pemesanan dan *tray* robot; *door\_status* untuk merekam status buka-tutup pintu; serta *users* yang menyimpan data autentikasi pengguna. Selain itu, terdapat juga tabel *images* untuk menyimpan metadata gambar seperti peta navigasi, tabel *jenis\_robot* untuk referensi klasifikasi robot, dan *koor* yang menyimpan data koordinat tujuan pengantaran. Seluruh data ini digunakan dan dimanipulasi oleh lapisan aplikasi yang berada di atasnya.

## 2. Application Layer (*Backend*)

Lapisan aplikasi terdiri dari dua komponen utama, yaitu *backend* PHP sebagai REST API server dan Node.js sebagai penyedia layanan *real-time* menggunakan WebSocket. *Backend* PHP bertugas mengelola seluruh logika bisnis inti sistem, termasuk proses autentikasi pengguna, validasi *input*, serta operasi CRUD (Create, Read, Update, Delete) terhadap entitas seperti baterai, pesanan, gambar, dan pengguna. File PHP seperti *login.php*, *battery\_data.php*, dan *orders.php* diakses melalui protokol HTTP dan memberikan respon dalam format JSON ke *Frontend*. *Backend* ini juga berfungsi sebagai penghubung antara *Frontend* dan basis data. Sementara itu, server Node.js menjalankan peran sebagai socket layer yang menangani komunikasi dua arah secara *real-time*. Melalui protokol Socket.IO, Node.js dapat menerima notifikasi dari *backend* PHP—misalnya saat status baterai diperbarui atau data order baru dan segera menyampaikan event tersebut ke klien tanpa perlu melakukan refresh halaman.

## 3. Presentation Layer (*Frontend*)

Lapisan presentasi merupakan antarmuka pengguna yang dibangun menggunakan Vue.js, yang bertugas menampilkan data serta menerima *input* dari pengguna secara interaktif. Sistem *Frontend* ini terhubung ke *backend* melalui REST API untuk mengambil data seperti informasi baterai. Selain itu, *frontend* juga terkoneksi dengan layanan Socket.IO dari Node.js untuk mendengarkan event *real-time*, seperti pembaruan status robot atau notifikasi penting lainnya. Komponen seperti auth.js berperan penting dalam mengatur alur autentikasi dan komunikasi data antara *Frontend* dan *backend*. Melalui pendekatan ini, pengguna dapat memperoleh pengalaman interaktif yang *responsif*, di mana data akan otomatis diperbarui di layar tanpa memerlukan interaksi manual ulang.

## B. Alur Kerja Sistem

Bagian ini menyajikan gambaran menyeluruh mengenai alur kerja sistem yang telah dikembangkan, dimulai dari proses awal hingga ke tahapan operasional yang lebih kompleks. Tujuannya adalah untuk memberikan pemahaman yang sistematis mengenai bagaimana sistem berfungsi secara keseluruhan, serta bagaimana interaksi antar komponen dilakukan.

### 1. Inisialisasi Sistem dan Autentikasi Pengguna

Tahapan awal dimulai dari proses autentikasi pengguna melalui halaman *login* yang disediakan pada sistem web monitoring. *User* memasukkan kredensial berupa *email* dan *password*, kemudian data tersebut dikirim melalui *endpoint POST /api/login* untuk diverifikasi oleh AuthController.

Apabila autentikasi berhasil, server akan menghasilkan token JWT (JSON Web Token) yang disisipkan pada *header Authorization* untuk setiap request selanjutnya. Sistem ini menjamin bahwa hanya pengguna yang telah terverifikasi yang dapat mengakses *endpoint* yang bersifat kritis (misalnya untuk menambahkan data robot atau mengubah status pesanan), sehingga mendukung prinsip keamanan berbasis *token-based authentication*.

### 2. Manajemen Data Robot dan Jenis Robot

Setelah *login*, pengguna dengan hak akses (admin) dapat menambahkan entitas baru berupa:

- Jenis Robot melalui *endpoint POST /api/jenis-robot*
- Robot Baru melalui *endpoint POST /api/battery-data*

Proses ini melibatkan validasi *input*, seperti nama robot, nomor seri, dan *jenis\_robot\_id*, serta pengecekan *foreign key* pada tabel *jenis\_robot*. Jika validasi berhasil, maka data robot disimpan dalam tabel *battery\_data* dengan nilai awal *battery\_level* = 100 dan *battery\_performace* = 100. Informasi ini penting untuk kebutuhan monitoring status baterai dan pemeliharaan robot.

Data jenis robot dapat diakses oleh pengguna melalui *endpoint GET /api/jenis-robot*. Sementara data seluruh robot ditampilkan melalui *GET /api/battery-data*, yang akan menghasilkan struktur data terorganisir berdasarkan jenis robot.

### **3. Monitoring Status Robot Secara *Real-time***

Setiap robot yang telah terdaftar akan dimonitor melalui *Dashboard* web yang menampilkan metrik berikut:

- Persentase baterai
- Performa baterai
- Waktu dan lokasi pengiriman

Fitur ini didukung oleh sistem *backend* PHP yang mengintegrasikan data dari *ESP32* (sebagai mikrokontroler) melalui protokol komunikasi HTTP dan ditampilkan pada *Frontend* menggunakan WebSocket berbasis Node.js.

Sistem notifikasi *real-time* bekerja melalui kelas `SocketNotificationService` yang mengirimkan event seperti:

- '`new_battery_data`' ketika data baterai baru dikirim
- '`robot_data_updated_from_php`' ketika status robot diperbarui
- '`new_order_created`' ketika pesanan baru dibuat

Teknologi ini meningkatkan efisiensi operasional dan memastikan administrator mendapatkan informasi secara instan tanpa perlu *refresh* manual.

### **4. Pelacakan Pesanan**

Web monitoring memungkinkan administrator melacak pesanan setelah barista membuat pesanan baru di aplikasi, melalui *endpoint* POST `/api/orders`. Setelah pesanan dibuat:

1. Sistem menyimpan data ke tabel `orders`.
2. Sistem mengirimkan notifikasi ke pengguna yang terhubung melalui WebSocket.

Riwayat pesanan juga dapat diakses melalui *endpoint*:

- GET `/api/orders` untuk semua pesanan
- GET `/api/orders/{id}` untuk Detail pesanan tertentu

### **5. Visualisasi dan Antarmuka Monitoring**

*Frontend* web menyediakan visualisasi status robot, pesanan, dan baterai dalam tampilan grafis. Tampilan ini menggunakan desain modern dengan pendekatan minimalis, fokus pada kemudahan penggunaan (*usability*) oleh administrator.

### **6. Validasi, Keamanan, dan Ketahanan Sistem**

Semua *endpoint* penting menerapkan validasi *input* yang ketat untuk mencegah:

- Masukan kosong atau format tidak sesuai
- Pelanggaran integritas data (seperti kesalahan *foreign key*)

- Akses tidak sah (unauthorized access)

Sistem juga dilindungi oleh:

- Middleware autentikasi JWT
- Pengaturan CORS pada index.php untuk membatasi asal permintaan
- Logging *error* dan pengelolaan kode *responses* HTTP

## 7. Monitoring Gambar dan Data Tambahan

Selain data robot dan pesanan, sistem juga memungkinkan pengambilan gambar atau media pendukung yang ditampilkan dalam UI. Gambar dikodekan dalam format base64 dan dikirim melalui *endpoint* GET /api/images

### 4.2.10 Data Layer (Database)

Berikut struktur dasar tabel yang digunakan dalam sub-sistem *backend*. Basis data yang digunakan adalah MySQL dengan nama database dbtes. Berikut adalah struktur tabel yang relevan.

#### A. battery\_data

Tabel battery\_data digunakan untuk menyimpan informasi level baterai robot secara periodik. Dalam konteks sistem web monitoring, tabel ini sangat krusial karena menjadi sumber data utama untuk menampilkan status daya robot secara *real-time*. Informasi ini dapat diakses oleh administrator melalui *Dashboard*, sehingga mereka dapat mengetahui kapan robot perlu dilakukan pengisian ulang. Selain itu, data historis baterai yang tersimpan juga dapat digunakan untuk analisis performa daya dan mendeteksi degradasi baterai dalam jangka panjang. Fitur monitoring baterai ini mendukung prinsip preventive maintenance dan efisiensi operasional robot pengantar.

<b>id</b>	<b>name</b>	<b>serial</b>	<b>battery_p</b>	<b>battery_level</b>	<b>timestamp</b>	<b>created_at</b>	<b>updated_at</b>	<b>jenis_robot_id</b>
3	Tifa	SN01	89	20	2025-07-03 13:...	2025-07-03 13:...	2025-07-06 16:...	2
4	Tifa	SN02	100	100	2025-07-04 04:...	2025-07-04 04:...	2025-07-04 04:...	2
5	Cobot 1	SN-01	100	100	2025-07-04 04:...	2025-07-04 04:...	2025-07-06 09:...	3
6	Cobot 2	SN-02	100	100	2025-07-06 16:...	2025-07-06 16:...	2025-07-06 16:...	3
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

**Gambar 4. 10 Tabel Database battery\_data**

#### B. images

Tabel images berfungsi untuk menyimpan referensi file gambar berupa denah lokasi atau peta pergerakan robot di lingkungan operasional, yaitu layout Tel-U Coffee. Dalam sistem web monitoring, tabel ini digunakan untuk menampilkan peta statis.

<b>id</b>	<b>image_name</b>	<b>image_data</b>	<b>uploaded_at</b>
1	map 1	BLOB	2025-03-12 18:05:16
2	map 2	BLOB	2025-03-12 18:05:32
3	map 3	BLOB	2025-04-28 05:19:34
4	map 4	BLOB	2025-04-28 05:20:37
NULL	NULL	NULL	NULL

**Gambar 4. 11 Tabel Database images**

#### C. jenis\_robot

Tabel yang ditampilkan merupakan representasi data dari berbagai jenis robot dalam sistem, dengan informasi identifikasi dan kapabilitas pemetaan (mapping). Kolom id bertindak sebagai penanda unik untuk setiap entitas robot, sementara kolom nama memberikan deskripsi jenis robot, seperti *Delivery Robot*, *Collaborative Robot (Cobot)*, dan *Autonomous Mobile Robot (AMR)*. Fitur penting lainnya ditunjukkan oleh kolom has\_maps, yang menggunakan nilai biner (1 untuk "ya", dan 0 untuk "tidak") guna menunjukkan apakah robot tersebut memiliki kemampuan atau data pemetaan lingkungan. Dari data yang tersedia, hanya *Delivery Robot* yang memiliki pemetaan aktif (has\_maps = 1), sementara dua lainnya belum memiliki.

<b>id</b>	<b>nama</b>	<b>has_maps</b>
2	Delivery Robot	1
3	Collaborative Robot (Cobot)	0
4	Autonomous Mobile Robot (AMR)	0
NULL	NULL	NULL

**Gambar 4. 12 Tabel Database jenis\_robot**

#### D. orders

Tabel ini menyimpan data terkait penugasan pengantaran oleh robot, yang menunjukkan hubungan antara robot, pesanan, dan lokasi tujuan. Kolom id adalah identifikasi unik tiap tugas, sementara robot\_id mengacu pada robot tertentu yang ditugaskan dalam hal ini, robot dengan ID 3. Kolom order\_number menyatakan urutan tugas atau nomor pesanan, dan tray\_position mengindikasikan posisi baki (*tray*) tempat item diletakkan, misalnya "Atas". Lokasi tujuan tercantum dalam kolom table\_number dan coordinates, di mana dalam entri ini pengantaran ditujukan ke *Meja 1* dengan koordinat (6.0, 1.0). Sedangkan status menampilkan nilai 1, yang kemungkinan menunjukkan bahwa tugas sedang berlangsung atau telah selesai. Tabel ini penting dalam sistem manajemen tugas berbasis robot karena memungkinkan pelacakan Detail setiap pengantaran secara kronologis, spasial, dan fungsional, serta dapat

digunakan untuk analisis performa dan efisiensi operasional robot dalam lingkungan layanan otomatis.

<b>id</b>	<b>robot_id</b>	<b>order_number</b>	<b>tray_position</b>	<b>table_number</b>	<b>coordinates</b>	<b>created_at</b>	<b>status</b>
30	3	1	Atas	Meja 1	[6.0,1.0]	2025-07-06 06:15:48	1

**Gambar 4. 13 Tabel Database orders**

#### E. *users*

Tabel yang ditampilkan berfungsi sebagai daftar akun pengguna sistem, khususnya untuk keperluan otentikasi dan manajemen akses. Kolom id adalah identifikasi unik setiap pengguna, sedangkan *username* menyimpan nama pengguna yang digunakan untuk *login* ke sistem. Kolom *password* berisi *hash* menggunakan metode Argon2ID, bukan dalam bentuk teks asli (*plaintext*). Kolom name mengindikasikan peran pengguna, dalam hal ini kedua entri tercatat sebagai Administrator. Informasi waktu pembuatan dan pembaruan akun dicatat secara Detail dalam kolom *created\_at* dan *updated\_at*. Data ini menunjukkan bahwa sistem memiliki mekanisme kontrol akses yang berbasis peran dan keamanan sandi terenkripsi, yang merupakan elemen penting dalam menjaga integritas dan keamanan sistem informasi, terutama dalam konteks pengelolaan robot atau sistem otomatisasi berbasis web atau *cloud*.

<b>id</b>	<b>username</b>	<b>password</b>	<b>name</b>	<b>created_at</b>	<b>updated_at</b>
2	admin	\$2y\$10\$WX5SIotfquh2E5R1RwRjzerw7TRonfx...	Administrator	2025-06-30 14:27:59	2025-06-30 14:27:59

**Gambar 4. 14 Tabel Database users**

#### F. *log\_history*

<b>id</b>	<b>order_id</b>	<b>robot_id</b>	<b>action_type</b>	<b>status_before</b>	<b>status_after</b>	<b>message</b>	<b>timestamp</b>
4	20	3	created	HULL	0	Pesanan baru dengan ID 20 pada Meja 1.	2025-07-03 13:36:0
5	21	3	created	HULL	0	Pesanan baru dengan ID 21 pada Meja 2.	2025-07-03 13:40:1
6	22	3	created	HULL	0	Pesanan baru dengan ID 22 pada Meja 1.	2025-07-03 14:05:4
7	20	3	completed	0	1	Pesanan ID 20 pada Meja 1 selesai diantarkan	2025-07-03 14:07:5

**Gambar 4. 15 Tabel Database log\_history**

Tabel *log\_history* memiliki fungsi utama sebagai pencatat jejak aktivitas robot secara kronologis dalam sistem. Fungsinya adalah untuk merekam setiap perubahan status, tindakan yang diambil oleh robot, serta waktu terjadinya aksi tersebut. Hal ini memungkinkan sistem memiliki histori lengkap atas seluruh proses yang dilakukan oleh robot, mulai dari menerima pesanan, bergerak, mengalami hambatan, hingga menyelesaikan tugas yang dapat diketahui melalui website. Hal ini membantu administrator untuk dapat melihat dan memantau aktifitas robot melalui web dan aplikasi yang telah diintegrasikan.

## G. notification\_history

Tabel notification\_history yang ditampilkan pada gambar merupakan bagian dari sistem monitoring atau pelaporan dalam suatu aplikasi atau sistem robotika. Tabel ini mencatat riwayat notifikasi yang dikirim oleh sistem terhadap kondisi atau status dari robot yang diidentifikasi melalui kolom robot\_id. Berdasarkan dua entri yang tersedia, dapat diketahui bahwa sistem mendekripsi dua jenis kondisi, yakni battery\_low dan performance\_warning, yang masing-masing diberi label warning dan info pada kolom level, menunjukkan tingkat urgensinya. Kolom title dan message berisi deskripsi singkat serta pesan rinci terkait notifikasi, yang kemungkinan besar ditujukan untuk ditampilkan pada antarmuka pengguna atau dikirim ke sistem lain. Terakhir, kolom timestamp mencatat waktu. Struktur tabel ini menunjukkan penerapan prinsip-prinsip rekam jejak (logging) dalam sistem berbasis data, yang penting untuk keperluan analisis performa, pemeliharaan, serta pelacakan kesalahan (debugging).

<b>id</b>	<b>robot_id</b>	<b>type</b>	<b>title</b>	<b>message</b>	<b>level</b>	<b>timestamp</b>
1	3	battery_low	Baterai Lemah	Baterai Tifa lemah (...	warning	2025-07-06...
2	3	performance_warning	Performa Baterai Per...	Performa baterai Ti...	info	2025-07-06...
NULL	NULL	NULL	NULL	NULL	NULL	NULL

**Gambar 4. 16 Tabel Database notification\_history**

### 4.2.11 Application Layer (Backend)

#### A. BatteryData.php

```
// Fungsi untuk membuat data robot dan status baterainya
public function createRobotData(): void
{ // Ambil dan decode input JSON
    $input = json_decode(file_get_contents('php://input'), true);
    if (json_last_error() !== JSON_ERROR_NONE || !is_array($input)) {
        $this->sendResponse(400, 'Invalid input data (malformed
JSON).');
        return;
    }
    // Validasi field wajib
    if (empty($input['name']) || empty($input['serial']) ||
empty($input['jenis_robot_id'])) {
        $this->sendResponse(400, 'Fields "name", "serial", and
"jenis_robot_id" are required.');
        return;
    }
    $idJenisRobot = (int) $input['jenis_robot_id'];
    if ($idJenisRobot <= 0) {
        $this->sendResponse(400, '"jenis_robot_id" must be a positive
integer.');
        return;
    }
    try {
        // Cek ID jenis robot
        $stmt = $this->pdo->prepare("SELECT id FROM jenis_robot WHERE
id = :jenis_robot_id");
        $stmt->execute(['jenis_robot_id' => $idJenisRobot]);
        $row = $stmt->fetch();
        if ($row) {
            $this->sendResponse(200, 'Robot found');
        } else {
            $this->sendResponse(404, 'Robot not found');
        }
    } catch (Exception $e) {
        $this->sendResponse(500, 'Internal server error');
    }
}
```

```

$stmt->execute([':jenis_robot_id' => $idJenisRobot]);
if (!$stmt->fetchColumn()) {
    $this->sendResponse(404, "Robot type with ID
{$idJenisRobot} not found.");
    return;
}
// Siapkan dan simpan data
$dataToCreate = [
    'name' => $input['name'],
    'serial' => $input['serial'],
    'battery_level' => 100,
    'battery_performace' => 100,
    'timestamp' => $input['timestamp'] ?? null,
    'jenis_robot_id' => $idJenisRobot,
];
$newBatteryDataId = $this->batteryDataModel-
>create($dataToCreate);
if ($newBatteryDataId === false) {
    $this->sendResponse(409, 'Failed to save robot/battery
data.');
    return;
}
// Ambil ulang data & kirim notifikasi
$createdData = $this->batteryDataModel->findById((int)
$newBatteryDataId);
if (!$createdData) {
    $this->sendResponse(500, 'Data saved but could not be
retrieved.');
    return;
}
if (!$this->socketNotifier->notify('new_battery_data',
$createdData)) {
    error_log("Warning: Failed to notify for battery data ID:
{$newBatteryDataId}");
}

```

Fungsi `createRobotData()` bertanggung jawab untuk menangani proses pembuatan data robot dan informasi baterainya melalui permintaan HTTP berbasis JSON. Fungsi ini dimulai dengan mengambil *input* dari body request menggunakan `file_get_contents('php://input')`, lalu menguraikannya menjadi array asosiatif melalui `json_decode`. Jika format JSON yang diterima tidak valid atau bukan array, maka fungsi langsung mengembalikan respon HTTP 400 dengan pesan bahwa data *input* tidak valid. Setelah format terverifikasi, fungsi memeriksa apakah tiga *field* penting, yaitu *name*, *serial*, dan *jenis\_robot\_id*, telah diisi. Jika salah satu dari *field* ini kosong, maka server akan mengembalikan pesan *error* bahwa ketiga *field* tersebut wajib diisi.

Setelah validasi awal, fungsi kemudian memeriksa apakah nilai *jenis\_robot\_id* yang diberikan merupakan bilangan bulat positif. Validasi ini penting karena id jenis robot digunakan sebagai foreign key untuk menghubungkan dengan tabel *jenis\_robot*. Fungsi selanjutnya melakukan *query* ke database untuk memeriksa apakah id jenis robot yang dimaksud memang tersedia. Jika tidak ditemukan, maka sistem akan *merespons* dengan kode

HTTP 404 untuk menunjukkan bahwa jenis robot tersebut tidak ada dalam sistem. Jika id valid dan ditemukan, maka proses dilanjutkan dengan menyiapkan data yang akan disimpan, yaitu nama, serial, level baterai awal sebesar 100, performa baterai awal sebesar 100, timestamp (jika disediakan), dan id jenis robot.

Setelah semua data disiapkan, fungsi menggunakan model batteryDataModel untuk menyimpan data ke database. Jika proses penyimpanan gagal, misalnya karena nomor serial yang dimasukkan sudah ada, maka fungsi akan *merespons* dengan kode HTTP 409 yang menandakan konflik data. Jika penyimpanan berhasil, fungsi kemudian mengambil kembali data yang baru disimpan menggunakan id-nya untuk memastikan bahwa data memang sudah masuk ke sistem. Bila pengambilan data ini gagal, maka fungsi memberikan respon *error* 500 sebagai tanda adanya kesalahan internal.

Apabila data berhasil diambil, maka langkah berikutnya adalah mengirimkan notifikasi *real-time* melalui Socket.IO dengan event bernama new\_battery\_data. Notifikasi ini memungkinkan sistem lain atau antarmuka pengguna menerima informasi bahwa ada entri data baru. Jika pengiriman notifikasi ini gagal, fungsi tetap melanjutkan eksekusi dan hanya mencatat kesalahan di log. Terakhir, jika semua proses berjalan sukses, fungsi akan memberikan respon HTTP 201 sebagai tanda bahwa data berhasil dibuat, beserta isi data yang baru ditambahkan.

Fungsi ini juga dilengkapi dengan mekanisme penanganan kesalahan melalui tiga blok try-catch. Kesalahan akibat argumen yang tidak valid ditangani oleh InvalidArgumentException, kesalahan database ditangani oleh PDOException, dan kesalahan lainnya yang tak terduga ditangani oleh blok Throwable. Semua jenis kesalahan ini dicatat dalam log dan dibalas dengan respon yang sesuai agar sistem tetap robust dan aman terhadap gangguan.

```
// Fungsi untuk mengambil semua data baterai yang dikelompokkan berdasarkan jenis robot
public function getAllBatteryData(): void
{
    try {
        // Ambil semua data dari tabel jenis_robot
        $queryJenisRobot = "SELECT * FROM jenis_robot";
        $stmtJenisRobot = $this->pdo->prepare($queryJenisRobot);
        $stmtJenisRobot->execute();
    } catch (PDOException $e) {
        // Tangani kesalahan database
    } catch (InvalidArgumentException $e) {
        // Tangani argumen yang tidak valid
    } catch (Throwable $e) {
        // Tangani kesalahan lainnya
        error_log($e->getMessage());
    }
}
```

```

$jenisRobotList = $stmtJenisRobot->fetchAll(PDO::FETCH_ASSOC);

    // Inisialisasi array hasil
    $result = [];

    // Iterasi setiap jenis robot
    foreach ($jenisRobotList as $jenisRobot) {
        $idJenisRobot = (int) $jenisRobot['id'];

        // Ambil data baterai yang terkait dengan jenis robot tertentu
        $queryBatteryData = "SELECT * FROM battery_data WHERE
jenis_robot_id = :jenis_robot_id";
        $stmtBatteryData = $this->pdo->prepare($queryBatteryData);
        $stmtBatteryData->execute([':jenis_robot_id' =>
$idJenisRobot]);
        $batteryDataList = $stmtBatteryData-
>fetchAll(PDO::FETCH_ASSOC);

        // Gabungkan jenis robot dan data baterainya ke dalam array
        hasil
        $result[] = [
            'robot_type' => $jenisRobot,
            'battery_data' => $batteryDataList,
        ];
    }

    // Kirim response sukses beserta data hasil pengelompokan
    $this->sendResponse(200, 'Battery data grouped by robot types
retrieved successfully.', ['data' => $result]);

} catch (PDOException $e) {
    // Tangani kesalahan database
    error_log("Database error during battery data retrieval: " . $e-
>getMessage());
    $this->sendResponse(500, 'Database error occurred while
retrieving data.');
} catch (Throwable $e) {

```

Fungsi getAllBatteryData() bertugas untuk mengambil seluruh data baterai yang dikelompokkan berdasarkan jenis robot dari database, dan mengirimkannya dalam bentuk *response* JSON. Proses dimulai dengan eksekusi *query* untuk mengambil semua data dari tabel *jenis\_robot*. Data ini digunakan sebagai acuan untuk mengetahui semua jenis robot yang

tersedia di sistem. Setelah *query* pertama berhasil dieksekusi, data hasilnya disimpan dalam bentuk array asosiatif yang disebut `$jenisRobotList`.

Selanjutnya, fungsi menginisialisasi array kosong `$result` yang akan diisi dengan data gabungan dari jenis robot dan masing-masing data baterainya. Melalui perulangan `foreach`, fungsi menelusuri setiap jenis robot yang telah diambil. Pada setiap iterasi, fungsi mengambil id jenis robot dan menjalankan *query* kedua untuk mengambil seluruh data dari tabel `battery_data` yang memiliki `jenis_robot_id` sesuai dengan id yang sedang diproses. Hasil dari *query* ini juga disimpan sebagai array asosiatif.

Setelah mendapatkan data baterai untuk tiap jenis robot, fungsi menyusun data dalam bentuk struktur array asosiatif yang terdiri dari dua bagian utama: informasi jenis robot (`robot_type`) dan data baterai yang berkaitan (`battery_data`). Struktur ini kemudian ditambahkan ke dalam array `$result`. Proses ini diulangi hingga semua jenis robot dan data baterainya berhasil diproses.

Jika seluruh proses berhasil tanpa hambatan, maka fungsi mengirim *response* HTTP 200 disertai pesan sukses dan data hasil pengelompokan tersebut. Namun, jika terjadi kesalahan saat menjalankan *query* database, maka fungsi akan masuk ke blok `catch` dan mencatat error ke dalam log, kemudian mengirim *response* HTTP 500 dengan pesan bahwa terjadi kesalahan database. Jika kesalahan lain yang tidak terduga terjadi, misalnya error pada proses pemrosesan data di memori, maka error tersebut ditangani oleh blok `Throwable` dan dicatat Detail trace-nya, serta dikembalikan *response* error internal server. Dengan pendekatan ini, fungsi memberikan jaminan bahwa data ditampilkan secara terstruktur dan aman dari gangguan yang tidak diinginkan.

```
// Fungsi untuk menambahkan jenis robot baru ke dalam sistem
public function addRobotType(): void
{
    // Cek apakah pengguna telah terautentikasi
    if (!$this->authenticatedUser) {
        $this->sendResponse(401, 'Otorisasi diperlukan untuk menambahkan
jenis robot.');
        return;
    }
    // Ambil dan decode data JSON dari request body
    $input = json_decode(file_get_contents('php://input'), true);
```

```

// Validasi format JSON
if ($json_last_error () !== JSON_ERROR_NONE || !is_array($input)) {
    $this->sendResponse(400, 'Invalid input data (malformed JSON).');
    return;
}

// Validasi apakah nama jenis robot diisi
if (empty($input['nama'])) {
    $this->sendResponse(400, 'Field "nama" is required for adding a
robot type.');
    return;
}

// Bersihkan spasi pada nama
$typeName = trim($input['nama']);
try {
    // Cek apakah nama jenis robot sudah ada di database
    $stmt = $this->pdo->prepare("SELECT id FROM jenis_robot WHERE nama
= :nama");
    $stmt->execute([':nama' => $typeName]);
    if ($stmt->fetchColumn()) {
        $this->sendResponse(409, "Robot type '{$typeName}' already
exists.");
        return;
    }
    // Masukkan jenis robot baru ke database
    $stmt = $this->pdo->prepare("INSERT INTO jenis_robot (nama) VALUES
(:nama)");
    $stmt->execute([':nama' => $typeName]);
    // Ambil ID dari data yang baru saja dimasukkan
    $newRobotTypeId = $this->pdo->lastInsertId();
    // Validasi apakah ID berhasil didapatkan
    if (!$newRobotTypeId) {
        $this->sendResponse(500, 'Failed to save robot type. No ID
returned.');
        return;
    }
    // Ambil kembali data jenis robot berdasarkan ID
    $stmt = $this->pdo->prepare("SELECT id, nama FROM jenis_robot WHERE
id = :id");
    $stmt->execute([':id' => $newRobotTypeId]);
    $createdRobotType = $stmt->fetch(PDO::FETCH_ASSOC);
    // Pastikan data berhasil diambil kembali
}

```

```

if (!$createdRobotType) {
    $this->sendResponse(500, 'Robot type saved but could not be
retrieved.');
    return;
}
// Kirim respon sukses beserta data jenis robot baru
$this->sendResponse(201, 'Robot type successfully added.',
['robot_type' => $createdRobotType]);

```

Fungsi `addRobotType()` bertujuan untuk menambahkan jenis robot baru ke dalam sistem melalui *endpoint* API, dengan memastikan prosesnya aman, tervalidasi, dan hanya bisa dilakukan oleh pengguna yang telah terautentikasi. Fungsi diawali dengan pemeriksaan otorisasi. Jika pengguna belum *login* atau tidak memiliki sesi otentikasi yang valid (diperiksa melalui properti `$this->authenticatedUser`), maka sistem akan langsung mengembalikan respon HTTP 401 dengan pesan bahwa otorisasi diperlukan untuk melanjutkan proses.

Setelah itu, fungsi mengambil data masukan dari body permintaan HTTP yang diharapkan dalam format JSON. Data ini di-decode menjadi array asosiatif. Jika format JSON tidak valid atau gagal didekod, maka fungsi akan mengembalikan *error* 400 dengan pesan bahwa data *input* tidak sesuai format. Langkah berikutnya adalah memeriksa apakah *field* nama telah diisi oleh pengguna. *Field* ini merupakan nama dari jenis robot yang ingin ditambahkan. Jika kosong, maka fungsi kembali membatalkan proses dan memberikan respon *error* 400.

Jika semua *input* valid, sistem kemudian memeriksa ke database untuk memastikan bahwa nama jenis robot yang dimasukkan belum ada sebelumnya. Hal ini dilakukan untuk mencegah duplikasi data. Apabila ditemukan bahwa nama tersebut sudah ada, maka fungsi akan mengembalikan respon konflik (HTTP 409), menandakan bahwa entri serupa sudah tercatat sebelumnya. Namun jika nama tersebut belum ada, maka sistem mengeksekusi perintah INSERT untuk menambahkan nama jenis robot ke dalam tabel `jenis_robot`.

Setelah data berhasil dimasukkan, fungsi menggunakan `lastInsertId()` untuk mendapatkan id dari jenis robot yang baru ditambahkan. Jika id ini tidak tersedia (misalnya karena kegagalan penyimpanan), maka sistem akan mengembalikan *error* 500. Namun jika id berhasil diperoleh, langkah selanjutnya adalah mengambil kembali data yang baru saja dimasukkan untuk memastikan bahwa penyimpanan telah sukses dan data bisa digunakan atau ditampilkan kembali. Data ini diambil kembali melalui SELECT berdasarkan id yang baru dibuat.

Jika data yang diambil tidak ditemukan (meskipun sudah disimpan), maka fungsi juga akan memberikan respon *error* 500. Namun jika semua langkah berjalan lancar, maka sistem akan mengirimkan respon sukses HTTP 201 (Created), disertai dengan data jenis robot yang baru ditambahkan. Untuk menjaga keandalan dan keamanan, fungsi juga menyertakan dua blok penanganan kesalahan: satu untuk *error* yang bersifat database (PDOException) dan satu lagi untuk *error* umum lainnya (Throwable). Dengan demikian, fungsi ini menjamin bahwa hanya data valid dan unik yang dapat masuk, serta seluruh proses berjalan aman dengan respon yang informatif.

## B. server.js

```
// ===== Simpan Notifikasi ke Database =====
// Fungsi untuk menyimpan data notifikasi ke tabel notification_history
async function saveNotificationToHistory(notification) {
    try {
        const { robotId, type, title, message, level } = notification;
        let timestamp = new Date().toISOString().slice(0, 19).replace("T", " ");
        await pool.query(
            `INSERT INTO notification_history (robot_id, type, title, message, level, timestamp) VALUES (?, ?, ?, ?, ?, ?)`,
            [robotId, type, title, message, level, timestamp]
        );
    } catch (error) {
        console.error ("Gagal menyimpan notifikasi:", error .message);
    }
}

// ===== Deteksi dan Kirim Notifikasi =====
// Fungsi utama yang mendeteksi kondisi baterai atau performa robot dan
// mengirim notifikasi real-time ke Frontend
function checkAndSendNotifications(robotData, socketInstance) {
    const robotId = robotData.id;
    const robotName = robotData.name || `Robot ${robotId}`;
    const currentPerformace = parseInt(robotData.battery_performace, 10);
    const notificationsToSend = [];

    if (robotData.battery_level <= 20) {
        const notif = {
```

```

        type: "battery_low",
        title: "Baterai Lemah",
        message: `Baterai ${robotName} lemah
        (${robotData.battery_level}%)`,
        robotId,
        robotName,
        level: "warning",
        timestamp: new Date().toISOString(),
    };
    notificationsToSend.push(notif);
    saveNotificationToHistory(notif);
}

notificationsToSend.forEach((notif) => {
    const payload = JSON.stringify(notif);
    if (socketInstance) socketInstance.emit("new_notification", payload);
    else io.emit("new_notification", payload);
});
}

// ===== Middleware Autentikasi WebSocket
=====

// Validasi JWT token dari klien WebSocket untuk memastikan hanya pengguna sah yang dapat terhubung
io.use((socket, next) => {
    const token = socket.handshake.auth.token;
    try {
        const decoded = jwt.verify(token,
process.env.JWT_SECRET_KEY_FROM_PHP);
        socket.userData = decoded.data;
        next();
    } catch (err) {
        next(new Error ("Token tidak valid"));
    }
});

// ===== Saat Klien Terkoneksi =====
// Mengirimkan respons awal dan menangani permintaan data saat klien WebSocket terhubung
io.on("connection", (socket) => {

```

```

socket.emit("connection_ack", {
  message: "Berhasil terhubung",
  userId: socket.userData?.userId,
});

socket.on("request_initial_data", async (request) => {
  const robotId = request.id;
  const robotData = await fetchRobotDetailData(robotId);
  checkAndSendNotifications(robotData, socket);
});

```

Bagian pertama adalah fungsi `saveNotificationToHistory`, yang bertugas untuk menyimpan data notifikasi ke dalam tabel `notification_history` pada database MySQL. Fungsi ini menerima objek `notification` yang berisi informasi penting seperti `robotId`, `type`, `title`, `message`, `level`, dan waktu kejadian (`timestamp`). Sebelum disimpan, `timestamp` dikonversi ke format YYYY-MM-DD HH:MM:SS agar sesuai dengan tipe data DATETIME di database. Jika terjadi kegagalan saat menyimpan, pesan kesalahan akan ditampilkan di `console`. Fungsi ini memastikan bahwa semua notifikasi yang dikirim ke pengguna juga tercatat untuk keperluan historis atau audit.

Selanjutnya adalah fungsi `checkAndSendNotifications`, yang merupakan komponen utama dalam sistem deteksi kondisi robot. Fungsi ini menerima data satu robot (`robotData`) dan socket klien (`socketInstance`) jika ada. Di dalamnya, sistem memeriksa apakah level baterai robot berada pada kondisi rendah ( $\leq 20\%$ ). Jika iya, maka sistem membentuk objek notifikasi dengan informasi seperti jenis notifikasi, pesan peringatan, dan waktu. Notifikasi tersebut disimpan ke database melalui `saveNotificationToHistory`, lalu dikirim secara *real-time* ke *Frontend* melalui event `Socket.IO` bernama `new_notification`. Pengiriman bisa ditujukan ke klien tertentu (jika ada `socketInstance`) atau secara global ke seluruh klien yang terhubung.

Untuk menjaga keamanan komunikasi *real-time*, server menggunakan middleware autentikasi WebSocket melalui `io.use()`. Middleware ini mengambil token JWT dari permintaan handshake klien dan memverifikasinya menggunakan secret key yang sama dengan *backend PHP*. Jika token valid, informasi pengguna disimpan di objek `socket.userData`, dan koneksi diizinkan. Jika token tidak valid, koneksi langsung ditolak. Proses ini memastikan bahwa hanya pengguna yang sudah *login* dan terautentikasi yang dapat terhubung ke server `Socket.IO`.

Terakhir, ketika klien berhasil terhubung, server akan mengeksekusi blok io.on("connection"). Pada saat ini, server mengirimkan *respons* awal ke klien berupa event connection\_ack yang menandakan koneksi sukses. Selanjutnya, jika klien mengirim permintaan data awal melalui event request\_initial\_data, server akan memanggil fungsi fetchRobotDetailData berdasarkan id robot yang diminta, lalu meneruskan data tersebut ke checkAndSendNotifications. Dengan begitu, pengguna akan langsung mendapatkan peringatan kondisi robot secara *real-time* segera setelah terhubung.

#### 4.2.12 Presentation Layer (*Frontend*)

##### A. Halaman Daftar Robot

```
<template>
    <!-- Menampilkan daftar robot per jenisnya -->
    <div v-else>
        <div v-for="(robotList, jenis) in groupedRobots" :key="jenis" class="mb-4">
            <!-- Menampilkan nama jenis robot sebagai heading -->
            <h2 class="h5 text-primary">{{ jenis }}</h2>
            <div class="row">
                <!-- Looping kartu robot per jenis -->
                <div v-for="robot in robotList" :key="robot.id" class="col-lg-4 col-md-6">
                    <!-- Komponen kartu robot yang punya 3 aksi: Detail, edit, hapus -->
                    <RobotListCard
                        :robot="robot"
                        @Detail-click="goToDetail(robot)"
                        @edit-click="editRobot(robot)"
                        @delete-click="confirmDelete(robot.id, robot.serial)" />
                </div>
            </div>
        </div>
    </div>

    <!-- Modal untuk mengedit data robot -->
    <EditRobotModal
        v-if="isEditModalVisible"
        :robot="robotToEdit"
        @close="isEditModalVisible = false"
        @save="saveRobotChanges"
    />

    <!-- Modal konfirmasi untuk hapus robot -->
    <ConfirmModal
        :show="showConfirmModal"
        @confirm="deleteRobot"
        @cancel="showConfirmModal = false"
    />

    <!-- Modal notifikasi informasi setelah penghapusan -->
    <ConfirmModal
        :show="showInfoModal"
```

```

        :message="infoMessage"
        @confirm="showInfoModal = false"
    />
</template>

// State reaktif untuk daftar robot
const robots = ref([]);
const isLoading = ref(true);
const error = ref(null);

// State untuk pengelolaan modal edit
const isEditModalVisible = ref(false);
const robotToEdit = ref(null);

// State untuk konfirmasi hapus
const showConfirmModal = ref(false);
const robotIdToDelete = ref(null);

// State untuk menampilkan info setelah aksi sukses
const showInfoModal = ref(false);
const infoMessage = ref("");

// Fungsi untuk mengambil daftar robot dari server
const fetchRobots = async () => {
    isLoading.value = true;
    try {
        robots.value = await fetchRobotsListAPI();
    } catch (err) {
        error.value = err.message;
    } finally {
        isLoading.value = false;
    }
};

// Navigasi ke halaman Detail robot dengan id tertentu
const goToDetail = (robot) => {
    router.push({ name: "robotDetail", params: { id: robot.id } });
};

// Ambil Detail robot sebelum ditampilkan dalam modal edit
const editRobot = async (robot) => {
    robotToEdit.value = await getBatteryDataDetailAPI(robot.id);
    isEditModalVisible.value = true;
};

// Simpan hasil perubahan edit dan perbarui daftar
const saveRobotChanges = async (updatedRobot) => {
    await updateRobotAPI(updatedRobot.id, updatedRobot);
    isEditModalVisible.value = false;
    await fetchRobots(); // refresh daftar robot
};

// Tampilkan modal konfirmasi sebelum hapus
const confirmDelete = (id) => {
    robotIdToDelete.value = id;
    showConfirmModal.value = true;
};

// Eksekusi penghapusan robot dan tampilkan info

```

```

const deleteRobot = async () => {
    await deleteRobotAPI(robotIdToDelete.value);
    showConfirmModal.value = false;
    infoMessage.value = "Robot berhasil dihapus.";
    showInfoModal.value = true;
    await fetchRobots(); // refresh daftar setelah hapus
};

</script>

```

Kode Vue.js di atas merupakan bagian dari komponen *Frontend* untuk menampilkan dan mengelola daftar robot. Komponen ini dibangun menggunakan pendekatan deklaratif dan modular dengan prinsip reaktif dari Vue 3 (<script setup>). Secara garis besar, fungsinya adalah untuk mengambil data robot dari *backend*, mengelompokkannya berdasarkan jenis, lalu menampilkannya dalam bentuk kartu menggunakan komponen *RobotListCard*. Setiap kartu memiliki tiga aksi utama: membuka Detail robot, mengedit data robot, dan menghapus robot.

Ketika pengguna memilih untuk mengedit robot, aplikasi akan mengambil Detail robot dari server menggunakan `getBatteryDataDetailAPI()` lalu memunculkan modal *EditRobotModal*. Setelah pengguna menyimpan perubahan, data diperbarui melalui `updateRobotAPI()`, modal ditutup, dan daftar robot disegarkan kembali dari *backend* agar data terbaru ditampilkan.

Sementara itu, jika pengguna memilih untuk menghapus robot, aplikasi akan menampilkan *ConfirmModal* untuk meminta konfirmasi. Setelah konfirmasi, proses penghapusan dijalankan melalui `deleteRobotAPI()`, dan informasi keberhasilan ditampilkan menggunakan modal info. Selain itu, daftar robot juga akan diperbarui kembali setelah penghapusan.

Kode ini juga menggunakan fitur `computed()` untuk mengelompokkan robot berdasarkan jenis, dan `ref()` untuk mengelola *state* reaktif seperti `isEditModalVisible`, `robotToEdit`, `showConfirmModal`, serta `showInfoModal`. Proses navigasi ke halaman Detail dilakukan dengan vue-router, menggunakan fungsi `router.push()`.

## B. Halaman Detail Robot

```

<template>
    <div class="container-fluid">

        <!-- Judul halaman yang menampilkan nama dan serial robot -->
        <h3>{{ robotDetailSocket.robotData.value?.name || "Memuat..." }} ({{ robotDetailSocket.robotData.value?.serial || "..." }})</h3>
    </div>
</template>

```

```

<!-- Indikator pemuatan data -->
<div v-if="robotDetailSocket.isLoading.value">Memuat data...</div>

<!-- Menampilkan pesan error jika data gagal dimuat -->
<div v-if="robotDetailSocket.error .value">{{ robotDetailSocket.error
.value }}</div>

<!-- Jika data tersedia, tampilkan informasi status robot -->
<div v-if="robotDetailSocket.robotData.value">

    <!-- Kartu status baterai -->
    <StatusCard
        title="Baterai"
        :value="`${robotDetailSocket.robotData.value.battery ?? 'N/A'}%`"
    />

    <!-- Kartu performa baterai dengan visual progress bar -->
    <StatusCard
        title="Performa Baterai"
        :value="`${robotDetailSocket.robotData.value.battery_performace
?? 'N/A'}%`"
        :progress="robotDetailSocket.robotData.value.battery_performace
|| 0"
        :progress-bar-
        class="getProgressBarColorClass(robotDetailSocket.robotData.value.battery
_performace)"
    />

    <!-- Kartu status waktu operasional -->
    <StatusCard
        title="Lama Beroperasi"
        :value="`${robotDetailSocket.robotData.value.operatingTime?.totalActiveDa
ys || 0} Hari`"
    />

    <!-- Galeri peta jika tersedia -->
    <MapGallery          v-if="robotDetailSocket.robotData.value.has_maps"
:images="dynamicMapImages" />

    <!-- Daftar sensor yang dimiliki robot -->

```

```

        <SensorList :sensors="robotDetailSocket.robotData.value?.sensors || dummySensors" />

        <!-- Log aktivitas robot (ringkasan 5 entri) -->
        <ActivityLog      title="Log      Robot"      :logs="displayedRobotLogs"
@viewAll="showRobotLogModal" />

        <!-- Log aktivitas sensor -->
        <ActivityLog      title="Log      Sensor"      :logs="displayedSensorLogs"
@viewAll="showSensorLogModal" />
    </div>

        <!-- Modal log lengkap robot -->
        <LogModal                               modalId="robotLogModal"
:logs="robotDetailSocket.allRobotLogs.value" ref="robotModalRef" />

        <!-- Modal log lengkap sensor -->
        <LogModal      modalId="sensorLogModal"      :logs="allSensorLogs"
ref="sensorModalRef" />
    </div>
</template>

// Variabel reaktif untuk menyimpan peta dan log sensor
const dynamicMapImages = ref([]);
const isImagesLoading = ref(true);
const imagesError = ref(null);

// Mengambil 5 log terakhir untuk tampilan ringkas
const displayedRobotLogs = computed(() =>
robotDetailSocket.allRobotLogs.value?.slice(0, 5) || []);
const displayedSensorLogs = computed(() => allSensorLogs.value.slice(0, 5));

// Fungsi async untuk memuat semua gambar peta
const fetchAllImages = async () => {
    isImagesLoading.value = true;
    try {
        const imagesData = await getAllImagesAPI();
        // Gambar disimpan dalam bentuk Base64
        dynamicMapImages.value = imagesData.map(img =>
`data:image/png;base64,${img.image_data}`);
    } catch (err) {

```

```

        imagesError.value = err.message;
    } finally {
        isImagesLoading.value = false;
    }
};

// Ketika halaman dimount, ambil data robot & gambar jika ada
onMounted(() => {
    const id = route.params.id;
    if (id) robotDetailSocket.fetchRobotDetail(id);
    if (robotDetailSocket.robotData.value?.has_maps) fetchAllImages();

    // Fungsi untuk menampilkan modal log
    const showRobotLogModal = () => robotLogModalInstance?.show();
    const showSensorLogModal = () => sensorLogModalInstance?.show();

    // Fungsi untuk memberi warna pada progress bar performa
    const getProgressBarColorClass = (p) => {
        if (p > 75) return "bg-success";
        if (p > 50) return "bg-warning";
        return "bg-danger";
    };

```

Komponen ini adalah tampilan Detail untuk satu robot dalam sistem monitoring berbasis Vue.js yang menampilkan data secara *real-time* melalui koneksi socket. Informasi yang ditampilkan mencakup nama dan serial robot, status baterai, performa baterai, lama waktu operasional, serta daftar sensor yang terpasang. Jika robot memiliki data peta, maka galeri gambar akan dimuat dari server menggunakan API dan ditampilkan dalam bentuk visual menggunakan komponen MapGallery. Selain itu, terdapat juga dua jenis log aktivitas, yaitu log robot dan log sensor, yang ditampilkan sebagian di halaman utama dan dapat diakses seluruhnya melalui modal Bootstrap yang diinisialisasi secara manual. Fungsi pemantauan ini didukung oleh fitur-fitur reaktif seperti ref, watch, dan onMounted yang memastikan data terus diperbarui secara langsung saat ID robot berubah atau saat terjadi perubahan dalam data. Komponen ini dirancang modular, sehingga informasi tersaji secara terpisah dan jelas melalui sub-komponen seperti StatusCard, SensorList, ActivityLog, dan LogModal, menjadikan

tampilan ini efisien untuk pengguna dalam melakukan pemantauan kondisi robot secara menyeluruh tanpa harus berpindah halaman.

Selain menampilkan data secara *real-time*, komponen ini juga dirancang untuk memberikan pengalaman pengguna yang informatif dan *responsif*. Saat data sedang dimuat, pengguna diberikan umpan balik visual melalui indikator pemuatan (*spinner*) dan pesan status. Jika terjadi kesalahan saat mengambil data dari server, seperti error koneksi atau otorisasi, pengguna akan langsung diberi tahu melalui pesan alert yang sesuai. Hal ini penting untuk menjaga kejelasan alur informasi dan menghindari kebingungan pengguna dalam menafsirkan kondisi halaman.

Setiap bagian dari informasi robot ditampilkan dalam bentuk komponen yang intuitif. Misalnya, komponen StatusCard tidak hanya menampilkan angka, tapi juga menambahkan progres visual dan ikon, sehingga memperkuat pemahaman secara visual. Untuk log aktivitas, pengguna diberikan akses cepat ke ringkasan (5 entri terbaru), namun tetap memiliki opsi untuk membuka modal guna melihat seluruh riwayat aktivitas. Ini memberikan keseimbangan antara informasi ringkas dan Detail mendalam, tanpa membebani antarmuka utama.

Penggunaan watch pada properti route.params.id memastikan bahwa setiap kali pengguna berpindah ke halaman robot yang berbeda, data akan diperbarui secara otomatis. Fungsi onUnmounted juga memastikan bahwa ketika pengguna keluar dari halaman ini, semua koneksi dan *resource* akan dibersihkan, sehingga performa aplikasi tetap optimal dan tidak terjadi kebocoran memori. Pendekatan ini menunjukkan bahwa komponen dibangun dengan memperhatikan aspek efisiensi, keandalan, dan keterjangkauan bagi pengembang maupun pengguna akhir.

## C. Halaman Metrik Robot

```
<!-- Menampilkan nama robot dari data metrik yang diterima dari server -->
<h3>Metrik {{ metricsSocket.robotData.value?.name || "Memuat..." }}</h3>

<!-- Jika data masih dimuat, tampilkan indikator pemuatan -->
<div v-if="metricsSocket.isLoading.value">Memuat...</div>

<!-- Jika terjadi kesalahan (network, auth, atau stream), tampilkan pesan error -->
<div v-else-if="errorMsg">{{ errorMsg }}</div>

<!-- Jika data robot kosong atau tidak ditemukan -->
<div v-else-if="!metricsSocket.robotData.value?.name">Data tidak ditemukan</div>
```

```

<!-- Jika data tersedia, tampilkan seluruh grafik metrik -->
<div v-if="metricsSocket.robotData.value?.name" class="row">
    <!-- Menggunakan komponen MetricDisplayCard untuk setiap grafik -->
    <MetricDisplayCard
        v-for="(c, i) in chartConfigs"
        :key="i"
        :title="c.title"
        :chart-data="c.data"
        :chart-labels="labels"
        :chart-options="getChartOptions(c.title)"
        :chart-type="c.type"
        :dataset-options="c.datasetOptions"
    />
</div>
</div>
// Import komponen kartu grafik
import MetricDisplayCard from
"@/components/metricComponent/MetricDisplayCard.vue";

// Import logic socket untuk pengambilan data real-time metrik robot
import { useRobotMetricsSocket } from "@/services/useRobotMetricSocket";

// Ambil parameter ID robot dari URL
const route = useRoute();

// Inisialisasi koneksi socket metrik
const metricsSocket = useRobotMetricsSocket();

// Buat label tanggal untuk sumbu X grafik, mencakup 6 hari ke belakang
dan "Hari Ini"
const labels = ref(
    Array.from({ length: 7 }, (_, i) => {
        const d = new Date();
        d.setDate(d.getDate() - (6 - i));
        return i === 6 ? "Hari Ini" : `${String(d.getDate()).padStart(2,
"0")}/${String(d.getMonth() + 1).padStart(2, "0")}`;
    })
);

// Konfigurasi setiap grafik yang akan ditampilkan, beserta datanya
const chartConfigs = [
    {
        title: "Aktivitas Robot (Hari)", // Menampilkan apakah robot aktif
        atau tidak setiap hari
        data: metricsSocket.lamaOperasiChartData.value,
        type: "bar"
    },
    {
        title: "Banyaknya Pesanan Per Hari", // Jumlah pesanan yang diterima
        robot setiap hari
        data: metricsSocket.pesananHarianChartData.value,
        type: "bar",
        datasetOptions: {
            backgroundColor: "rgba(54,162,235,0.6)",
            borderColor: "rgb(54,162,235)"
        }
    },
    {

```

```

        title: "Performa Baterai (%)", // Menampilkan performa baterai dari
        waktu ke waktu
        data: metricsSocket.performaRobotChartData.value,
        type: "line"
    }
];

// Fungsi untuk mengatur konfigurasi grafik berdasarkan judul metrik
const getChartOptions = (title) => {
    const opt = {
        responsive: true,
        maintainAspectRatio: false,
        plugins: {
            legend: { display: false },
            title: { display: true, text: title },
            tooltip: { enabled: true }
        },
        scales: {
            y: { beginAtZero: true },
            x: {}
        }
    };

    // Khusus untuk metrik aktivitas, tampilkan skala 0 dan 1 dengan label
    // Aktif/Tidak Aktif
    if (title === "Aktivitas Robot (Hari)") {
        opt.scales.y.max = 1;
        opt.plugins.tooltip.callbacks = {
            label: ctx => ctx.parsed.y ? "Aktif" : "Tidak Aktif"
        };
    }

    // Khusus untuk performa baterai, batas atas 100% dan tooltip persentase
    if (title === "Performa Baterai (%)") {
        opt.scales.y.max = 100;
        opt.plugins.tooltip.callbacks = {
            label: ctx => `${ctx.parsed.y?.toFixed(0)}%`
        };
    }
}

```

Komponen ini menggunakan pendekatan `<script setup>`, yaitu sintaks *composition API* modern yang menawarkan penyederhanaan deklaratif, sekaligus meningkatkan keterbacaan dan efisiensi pengembangan. Struktur kode ini mencerminkan pemisahan tanggung jawab (*separation of concerns*), di mana setiap bagian dari tampilan ditangani oleh fungsi-fungsi spesifik yang mengatur pengambilan data, logika pemrosesan, dan pemetaan data ke komponen presentasi.

Data metrik diambil melalui koneksi *web socket* yang dibungkus oleh fungsi `useRobotMetricsSocket()` sebuah *composable* khusus yang bertugas mengatur status koneksi, pemrosesan data mentah, serta memfasilitasi *binding* dua arah terhadap data visualisasi. Data yang diterima dari *backend* kemudian dibagi menjadi tiga aspek utama, yaitu aktivitas harian robot, jumlah pesanan per hari, dan performa baterai dalam persen. Masing-masing aspek ini

diwakili dalam bentuk konfigurasi grafik melalui objek chartConfigs, yang berisi properti seperti *title*, *data*, *type*, dan *datasetOptions*. Pola ini menunjukkan bahwa komponen mendukung konfigurasi dinamis terhadap data yang akan divisualisasikan, mempermudah ekspansi bila di masa depan dibutuhkan metrik tambahan.

Elemen grafik yang digunakan dikelola oleh komponen anak bernama MetricDisplayCard, yang memanfaatkan pustaka chart untuk merender grafik. Komponen tersebut menerima data label waktu (7 hari terakhir), data nilai metrik, jenis grafik (*bar* atau *line*), serta opsi visualisasi khusus. Label waktu disusun secara dinamis melalui fungsi pembentuk *labels*, yang akan menghasilkan rentang waktu dalam format tanggal pendek dengan posisi terakhir selalu bertuliskan "Hari Ini". Strategi ini mempermudah pengguna dalam mengenali konteks temporal dari data yang disajikan.

Salah satu aspek penting dari kode ini adalah adanya fungsi getChartOptions, yang bertugas untuk menyesuaikan konfigurasi *chart* berdasarkan konteks judul grafik. Fungsi ini tidak hanya menetapkan opsi visual dasar seperti grid dan skala sumbu, tetapi juga melakukan *custom tooltip* agar informasi yang ditampilkan lebih mudah dipahami oleh pengguna awam. Sebagai contoh, untuk grafik "Aktivitas Robot (Hari)", nilai 1 ditafsirkan sebagai "Aktif", sedangkan 0 ditampilkan sebagai "Tidak Aktif" pendekatan ini memberikan makna semantik yang kuat dibandingkan hanya menampilkan angka mentah.

Dalam hal pemrosesan dan *lifecycle*, komponen ini memanfaatkan fungsi *onMounted* dan *watch* untuk memastikan bahwa data metrik akan selalu dimuat atau diperbarui setiap kali terjadi perubahan pada parameter *id* yang diambil dari rute URL. Pendekatan reaktif ini menjamin bahwa data yang ditampilkan selalu selaras dengan identitas robot yang sedang dipantau. Tidak kalah penting, *onUnmounted* digunakan untuk memanggil metode *cleanup* dari *metricsSocket*, yang menandakan adanya praktik manajemen sumber daya (*resource cleanup*) guna menghindari kebocoran memori akibat socket yang tidak tertutup.

## D. Halaman Tambah Robot

```
<template>
<div class="container mt-4">
  <div class="card shadow-sm">
    <div class="card-header">
      <h3 class="card-title mb-0">Tambah Robot Baru</h3>
    </div>
    <div class="card-body">
      <form @submit.prevent="submitForm">
        <div class="mb-3">
          <label for="namaRobot" class="form-label">Nama Robot</label>
```

```

<input
  type="text"
  class="form-control"
  id="namaRobot"
  v-model="robotData.name"
  required
/>
</div>

<div class="mb-3">
  <label for="serialRobot" class="form-label">Nomor Seri</label>
  <input
    type="text"
    class="form-control"
    id="serialRobot"
    v-model="robotData.serial"
    required
  />
</div>

<div class="mb-3">
  <label for="jenisRobot" class="form-label">Jenis Robot</label>
  <select
    class="form-select"
    id="jenisRobot"
    v-model="robotData.jenis_robot_id"
    required
  >
    <option value="" disabled>Pilih Jenis Robot...</option>
    <option
      v-for="jenis in jenisRobotList"
      :key="jenis.id"
      :value="jenis.id"
    >
      {{ jenis.nama }}
    </option>
  </select>
</div>

<div class="d-flex justify-content-end">
  <button type="submit" class="btn btn-primary" :disabled="isSubmitting">
    {{ isSubmitting ? 'Menyimpan...' : 'Simpan' }}
  </button>
</div>
</form>
</div>
</div>

<div v-if="showSuccessMessage" class="alert alert-success mt-4 text-center">
  <strong>{{ successRobotName }}</strong>&nbsp;berhasil ditambahkan!
</div>
</div>
</template>

<script setup>
const robotData = ref({ name: "", serial: "", jenis_robot_id: "" });
const jenisRobotList = ref([]);
```

```

const isSubmitting = ref(false);
const showSuccessMessage = ref(false);
const successRobotName = ref('');

const fetchJenisRobot = async () => {
  try {
    jenisRobotList.value = await getRobotTypesAPI();
  } catch (err) {
    alert("Gagal memuat jenis robot.");
  }
};

onMounted(fetchJenisRobot);

const submitForm = async () => {
  if (!robotData.value.jenis_robot_id) {
    alert("Silakan pilih jenis robot.");
    return;
  }

  isSubmitting.value = true;
  showSuccessMessage.value = false;

  try {
    const payload = {
      name: robotData.value.name,
      serial: robotData.value.serial,
      jenis_robot_id: parseInt(robotData.value.jenis_robot_id),
    };
    const createdRobot = await createRobotDataAPI(payload);

    successRobotName.value = createdRobot.name;
    showSuccessMessage.value = true;

    robotData.value = { name: "", serial: "", jenis_robot_id: "" };
  } catch (err) {
    alert("Gagal menyimpan robot.");
  } finally {
    isSubmitting.value = false;
  }
};
</script>

```

Kode ini merupakan komponen Vue 3 yang digunakan untuk menambahkan data robot baru ke dalam sistem. Komponen ini terdiri dari *form* yang memungkinkan pengguna mengisi nama robot, nomor seri, dan jenis robot yang diambil dari API. Ketika halaman dimuat (*onMounted*), aplikasi akan memanggil API *getRobotTypesAPI()* untuk memuat daftar jenis robot yang tersedia ke dalam *dropdown*. Hal ini memungkinkan pengguna memilih jenis robot yang relevan.

Saat *form* dikirim, fungsi *submitForm* akan dijalankan. Fungsi ini terlebih dahulu memvalidasi apakah jenis robot telah dipilih. Jika valid, maka data dikemas dalam bentuk objek *payload* dan dikirim ke *backend* melalui API *createRobotDataAPI()*. Jika penyimpanan

berhasil, nama robot yang berhasil ditambahkan akan ditampilkan melalui pesan sukses di bawah *form*. Komponen ini juga menangani status *loading* dengan boolean *isSubmitting*, serta membersihkan data *form* setelah proses selesai.

Secara keseluruhan, kode ini memfasilitasi proses pembuatan entri data robot baru secara efisien dan terintegrasi dengan sistem *backend*, serta memberikan *feedback real-time* kepada pengguna ketika aksi berhasil dilakukan.

#### 4.2.13 Hasil Implementasi

Sistem web monitoring TIFA berhasil diimplementasikan dengan *backend* PHP dan *Frontend* Vue.js yang terintegrasi secara efektif. *Backend* mampu melayani permintaan data baterai, performa baterai, dan status robot dengan validasi *input* yang baik serta *respons* JSON konsisten. *Frontend* menampilkan informasi *real-time* seperti status baterai dan daftar pesanan aktif secara *responsif* dan interaktif.

Komunikasi data antara *Frontend* dan *backend* menggunakan protokol HTTP dengan format JSON berjalan lancar, memungkinkan pembaruan data tanpa perlu *refresh* halaman. Sistem juga dilengkapi mekanisme *debugging* memudahkan pengembang untuk menemukan letak kesalahan teknis pada website serta web juga dilengkapi dengan fitur *login* sebagai fitur keamanan.

#### 4.2.14 Penggunaan *Hardware* Untuk Web

Berikut merupakan penggunaan *hardware* untuk web monitoring.

**Tabel 4. 2 Penggunaan *Hardware* Web Monitoring**

Komponen	Spesifikasi
Jenis Layanan	Instance Deployment
<i>Backend Server</i>	PHP BE Server
<i>Frontend Server</i>	Vue.js FE
Spesifikasi Instance	- 1 vCPU - 1GB RAM - 25GB SSD - Ubuntu 22.04 LTS
Fitur Tambahan	- Deploy CI/CD - UFW/Firewall - SSL - Region: Sg/Eu

#### **4.2.15 Detil Implementasi Monitoring Sensor IR ke UI**

Sistem terdistribusi ini terdiri dari dua bagian pengembangan oleh divisi berbeda. Divisi Mekatronik bertanggung jawab atas pembacaan 5 sensor Infrared (IR) dan pengiriman data ke ESP32 utama melalui Arduino. Divisi IoT bertugas melakukan pemrosesan data, komunikasi jaringan, penyimpanan, dan visualisasi UI. Sistem ini menerapkan arsitektur terdistribusi mulai dari pembacaan sensor hingga visualisasi data pada antarmuka pengguna untuk memastikan tombol konfirmasi hanya aktif ketika semua pesanan telah diambil. Fokus utama implementasi adalah pada layer komunikasi dari ESP32 ke UI melalui *backend* dan database.

#### **4.2.16 Layer Pemrosesan Status Konfirmasi pada *Backend PHP***

Sistem ini dirancang untuk mengontrol pintu otomatis berdasarkan data dari lima sensor IR yang dikirim dari Arduino ke ESP32. Pintu akan ditutup ketika semua sensor IR bernilai 0, yang menunjukkan bahwa *tray* kosong. Kondisi ini dirumuskan sebagai berikut:

$$Tray\ Kosong = \sum_{i=1}^5 IR_i = 0$$

Persamaan menyatakan bahwa pintu akan menutup otomatis jika semua sensor IR bernilai 0, yang mengindikasikan *tray* kosong. Kondisi ini sejalan dengan logika AND, di mana setiap sensor harus memenuhi  $IR_i=0$ . Sistem ini memastikan pintu hanya terbuka saat minimal satu sensor mendeteksi objek dan segera tertutup kembali saat *tray* kosong, sehingga mencegah akses tidak diinginkan diinginkan.

Fungsi:

- A. Menerima dan memproses data array dari lima sensor IR yang dikirim oleh Arduino ke ESP32.
- B. Menghitung jumlah nilai sensor untuk menentukan status *tray* (kosong atau berisi).
- C. Mengontrol pintu otomatis: pintu tertutup ketika *tray* kosong (semua sensor bernilai 0) dan terbuka ketika ada objek yang terdeteksi.

ESP32 bertugas menerima data sensor, memprosesnya, dan mengontrol pintu menggunakan pin yang terhubung ke *tray*. Berikut adalah implementasi kode pada ESP32

```

void loop() {
    // Kirim status IR ke ESP32 setiap 500ms
    if (millis() - waktuCekIR >= 500) {
        waktuCekIR = millis();
        int ir1 = digitalRead(IR1);
        int ir2 = digitalRead(IR2);
        int ir3 = digitalRead(IR3);
        int ir4 = digitalRead(IR4);
        int ir5 = digitalRead(IR5);
        bool semuaHIGH = (ir1 == HIGH && ir2 == HIGH && ir3 == HIGH &&
ir4 == HIGH && ir5 == HIGH);
        if (semuaHIGH && !statusIRsebelumnya) {
            Serial2.println("MAKANAN_TIDAK_ADA");
            Serial.println("[Arduino] Kirim: MAKANAN_TIDAK_ADA");
            statusIRsebelumnya = true;
        } else if (!semuaHIGH && statusIRsebelumnya) {
            Serial2.println("MAKANAN_ADA");
            Serial.println("[Arduino] Kirim: MAKANAN_ADA");
            statusIRsebelumnya = false;}}

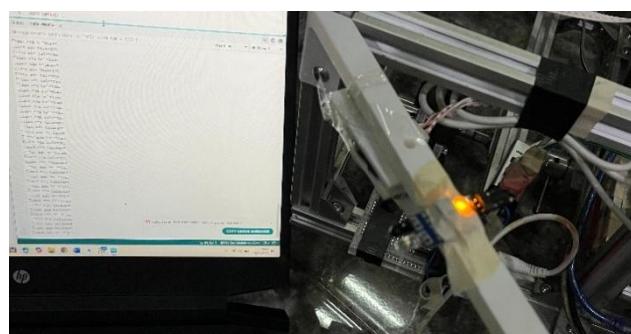
```

Kode ini memastikan bahwa pintu otomatis tertutup ketika semua sensor IR bernilai 0 (*tray kosong*) dan terbuka ketika ada objek yang terdeteksi oleh salah satu sensor.



**Gambar 4. 17 IR high**

Kondisi IR sensor 1 (HIGH): Pesanan terdeteksi.



**Gambar 4. 18 IR low**

Kondisi IR sensor 0 (LOW): *Tray kosong* (pesanan sudah diambil).

## 4.3 Prosedur Pengoperasian

Berikut adalah prosedur pengoperasian untuk aplikasi dan web TIFA.

### 4.3.1 Prosedur Pengoperasian untuk Aplikasi TIFA

Tel-U Interactive Food Assistant (TIFA) merupakan sistem aplikasi *mobile* yang berfungsi sebagai antarmuka monitoring dan pengendalian robot pengantar makanan di Tel-U Coffee. Aplikasi ini diimplementasikan untuk perangkat Android berbasis tablet yang terhubung dengan sistem *backend* dan mikrokontroler robot secara *real-time*. Antarmuka aplikasi dirancang *responsif* menggunakan Jetpack Compose dengan tema modern yang mengedepankan kemudahan penggunaan bagi barista dan staf operasional Tel-U Coffee [53]

#### A. Menu Utama

Setelah membuka aplikasi TIFA, pengguna akan diarahkan ke Menu Utama dengan fitur-fitur utama sebagai berikut:



Gambar 4. 19 Halaman Utama

##### 1. Fitur Antar:

- Digunakan untuk mengelola dan memulai proses pengantaran pesanan oleh robot.
- Pengguna dapat memilih tujuan pengantaran dan memantau prosesnya.
- Merupakan fitur inti dari sistem robot pengantar ini.

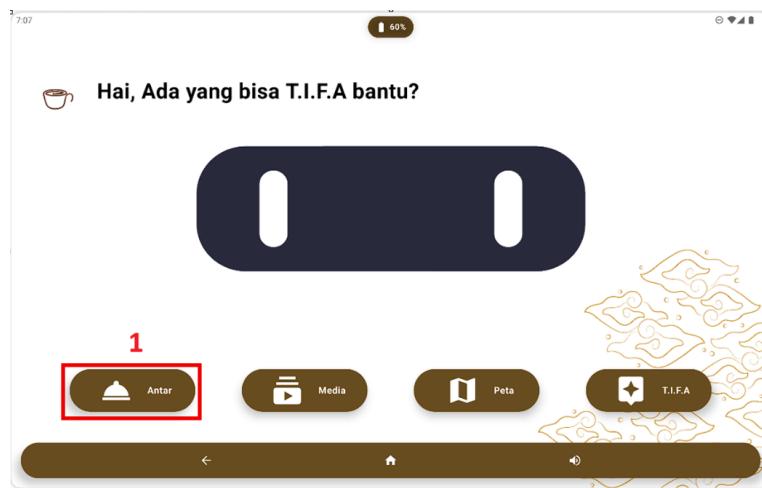
##### 2. Fitur Media:

- Memberikan akses untuk mengatur konten media (seperti Foto dan Video) yang akan ditampilkan oleh robot saat melakukan pengantaran.

- Cocok untuk menyisipkan informasi promosi, ucapan, atau konten visual lainnya.
3. Fitur Peta: Mengarahkan pengguna ke tampilan peta yang menunjukkan area pengantaran robot yang tersimpan.
  4. Fitur *TalkBack*: Menyediakan suara tentang informasi T.I.F.A.

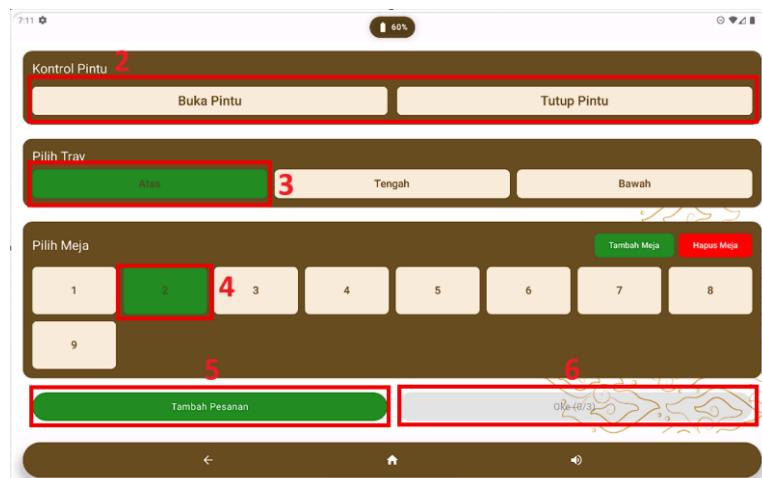
## B. Mode Antar

Langkah-langkah menggunakan mode antar:



**Gambar 4. 20 Halaman Utama**

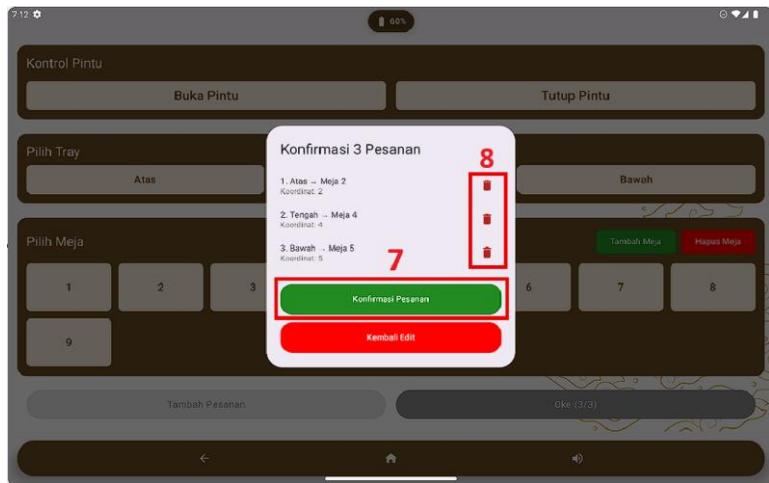
1. Tekan tombol Antar pada aplikasi.



**Gambar 4. 21 Halaman Antar**

2. Gunakan tombol Buka Pintu dan Tutup Pintu untuk memasukkan pesanan ke robot.
3. Pilih *tray* yang akan dipakai (Atas, Tengah, Bawah).
4. Pilih nomor meja tujuan pengantaran.

5. Tekan Tambah Pesanan untuk menambahkan pesanan (maksimal 3 pesanan).
6. Tekan Oke jika sudah sesuai.



**Gambar 4. 22 Pop up Halaman Antar**

7. Tekan Konfirmasi Pesanan untuk mengirim perintah pengantaran ke robot.

Langkah opsional:

8. Jika Anda melakukan kesalahan *input*, tekan ikon sampah untuk menghapus.

### C. Pengelolaan Meja

Langkah menambahkan dan menghapus meja:

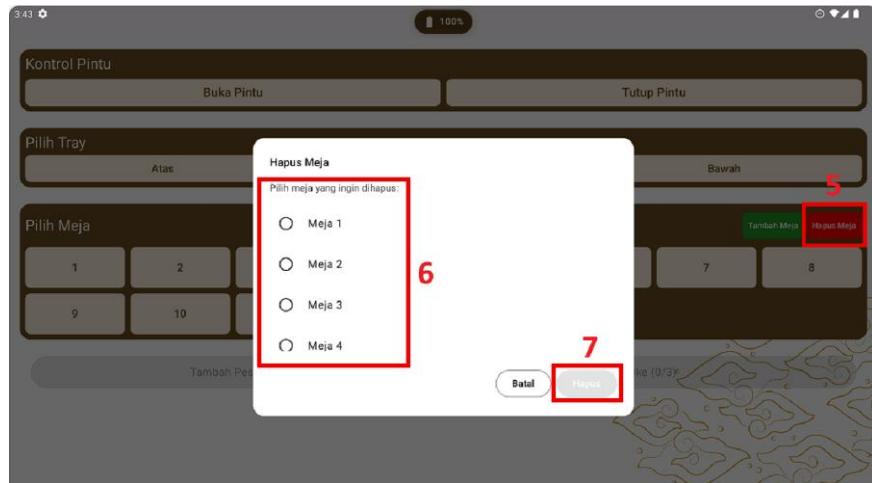


**Gambar 4. 23 Pop-up Tambah Meja**

Menambah meja:

1. Tekan tombol Tambah Meja.
2. Isi kolom Nomor Meja ketik nama atau nomor meja, contoh: Meja 11.
3. Isi kolom Koordinat masukkan angka koordinat meja, contoh: 11. Angka ini akan digunakan robot untuk navigasi ke meja tersebut.

4. Tekan tombol hijau Tambah tombol ini akan mengirim data ke server untuk disimpan ke database. Setelah berhasil ditambahkan, meja akan muncul pada daftar di bawah bagian Pilih Meja.

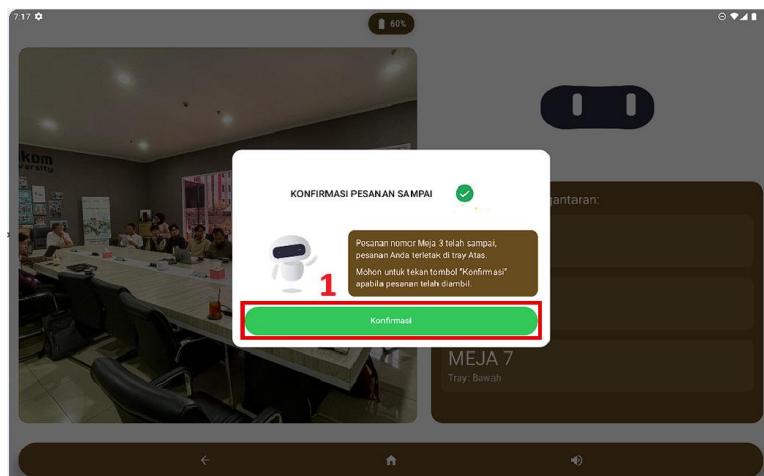


**Gambar 4. 24 Pop-up Hapus Meja**

Menghapus meja (opsional):

5. Jika ingin menghapus nomor meja, tekan tombol Hapus Meja.
6. Pilih meja yang ingin dihapus dari daftar.
7. Tekan tombol Hapus, Meja akan terhapus dari daftar.

#### D. Konfirmasi Pengambilan Pesanan



**Gambar 4. 25 Pop-up Konfirmasi**

Langkah:

1. Setelah robot sampai di meja, pelanggan mengambil pesanan lalu menekan tombol Konfirmasi pada aplikasi untuk menandai pesanan sudah diterima.

## E. Fitur Media



Gambar 4. 26 Halaman Utama

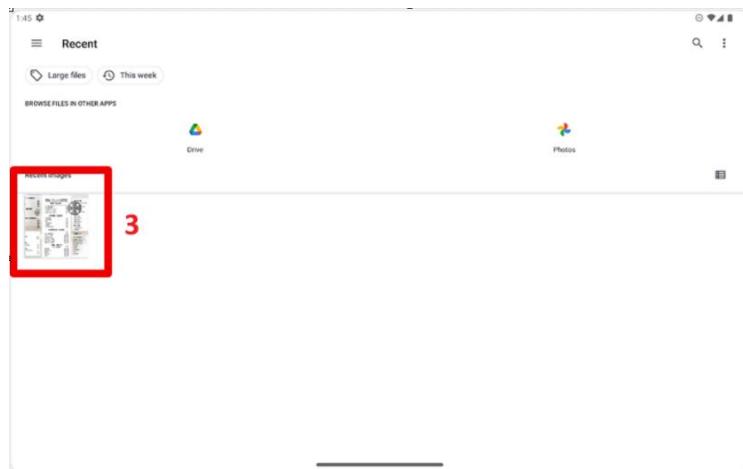
Langkah menggunakan fitur media:

1. Tekan tombol Media di halaman utama.



Gambar 4. 27 Halaman Media

2. Pilih fitur di Tambah Foto, Tambah Video.



**Gambar 4. 28 Penyimpanan Media**

3. Pilih Foto yang ingin di tambah.

Langkah opsional:

4. Jika ingin menghapus media terakhir maka tekan tombol Hapus Item.

## F. Fitur Peta



**Gambar 4. 29 Halaman Utama**

1. Tekan tombol Peta di halaman utama, maka langsung bisa melihat peta pada saat pindah ke halaman peta.

## G. Fitur TalkBack



**Gambar 4. 30 Halaman Utama**

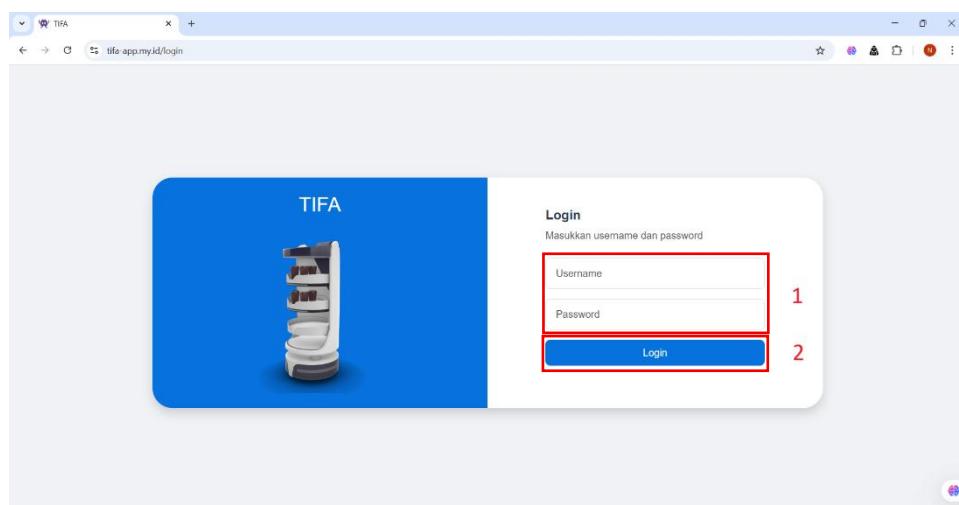
1. Tekan tombol T.I.F.A, maka fitur *TalkBack* akan muncul.

### 4.3.2 Prosedur Pengoperasian untuk Web Monitoring

Berikut merupakan uraian prosedur pengoperasian untuk web monitoring TIFA.

#### A. Login

Apabila web monitoring TIFA telah diakses melalui *browser*, halaman pertama yang akan muncul yaitu halaman *login*. *User* dapat melakukan *login* dengan akun yang sudah terdaftar.

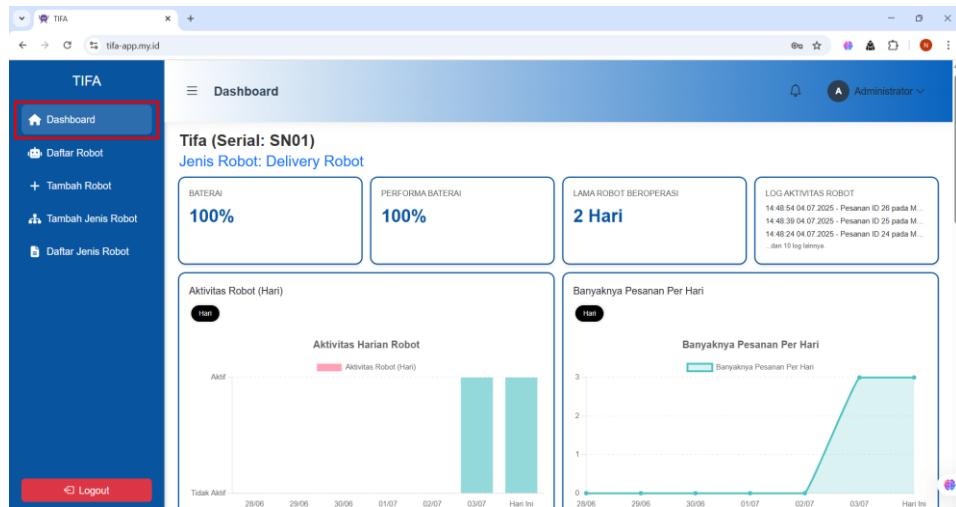


**Gambar 4. 31 Halaman Login**

1. Masukkan *username* dan *password* yang telah terdaftar.
2. Klik tombol “*Login*”.

## B. Dashboard

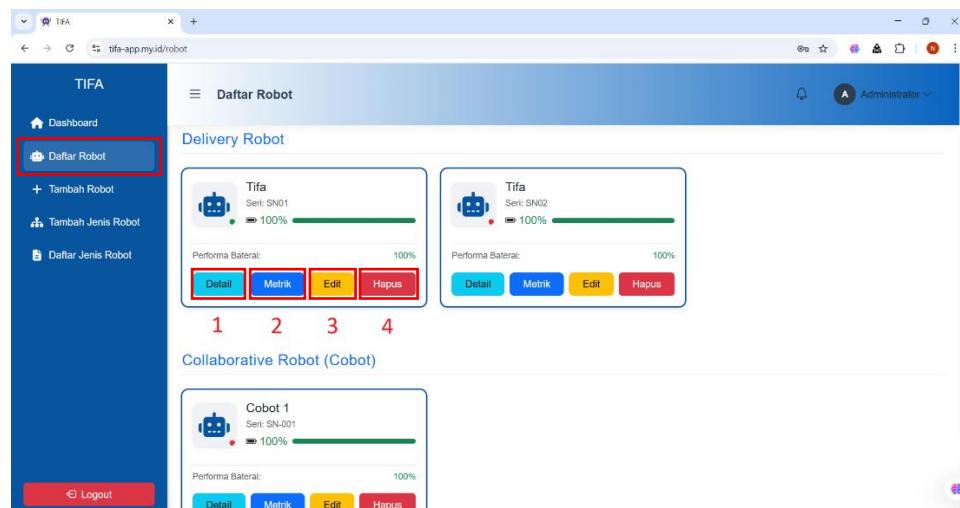
Setelah klik tombol “Login” user diarahkan menuju halaman *dashboard* yang menampilkan sekilas informasi robot secara *real-time* untuk semua robot yang terdaftar. Lalu user dapat memilih beberapa menu yang tersedia pada *sidebar* sesuai kebutuhan.



Gambar 4. 32 Halaman *Dashboard*

## C. Daftar Robot

Pada halaman daftar robot disajikan *list* semua robot yang terdaftar.

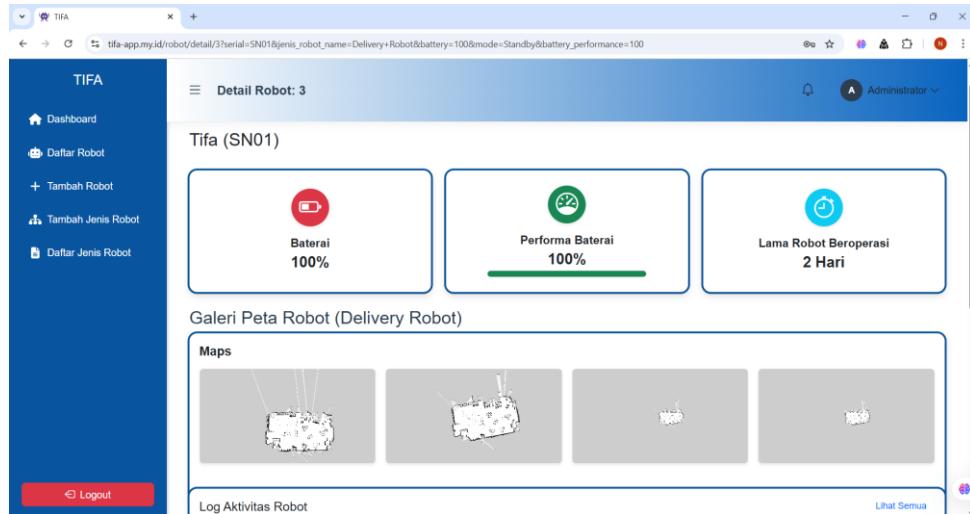


Gambar 4. 33 Halaman Daftar Robot

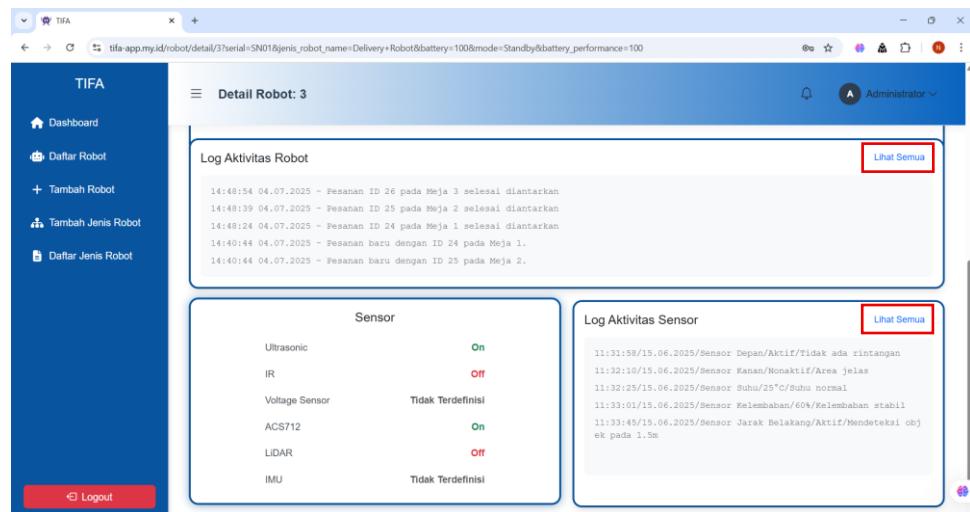
1. Untuk memonitoring detail robot secara rinci setiap robot, klik tombol “Detail”.
2. Untuk memonitoring informasi mengenai robot yang divisualisasikan menggunakan grafik, klik tombol “Metrik”.
3. Untuk edit nomor seri robot dan jenis robot, klik tombol “Edit”.
4. Untuk hapus robot beserta semua data yang berkaitan, klik tombol “Hapus”.

## D. Detail Robot

Setelah klik tombol “Detail” *user* diarahkan menuju halaman detail robot yang menampilkan informasi robot secara rinci dan *real-time* untuk satu robot. Pada halaman ini terdapat informasi mengenai persentase baterai robot, performa baterai, lama robot beroperasi, *maps*, log aktivitas robot, status sensor, serta log aktivitas sensor.

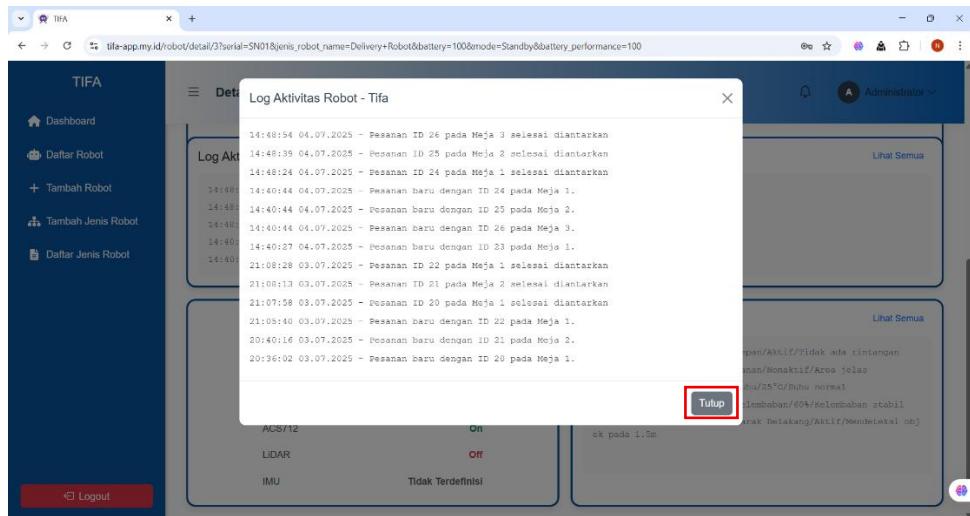


Gambar 4. 34 Halaman Detail Robot (1)

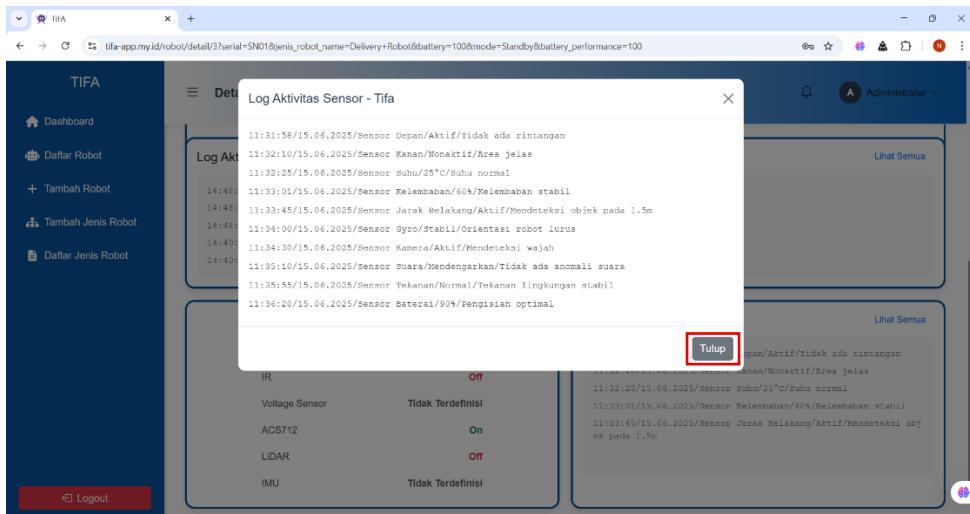


Gambar 4. 35 Halaman Detail Robot (2)

Klik tombol “Lihat Semua” untuk membuka *pop-up* yang berisi semua data log aktivitas robot dan log aktivitas sensor.



**Gambar 4. 36 Pop-up Log Aktivitas Robot**

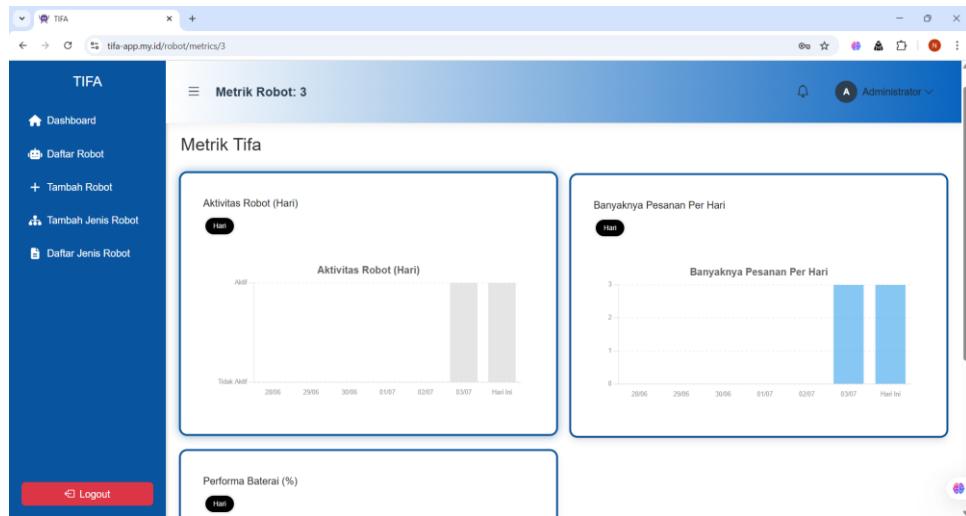


**Gambar 4. 37 Pop-up Log Aktivitas Sensor**

Untuk menutup *pop-up*, *user* dapat klik tombol “Tutup”.

## E. Metrik

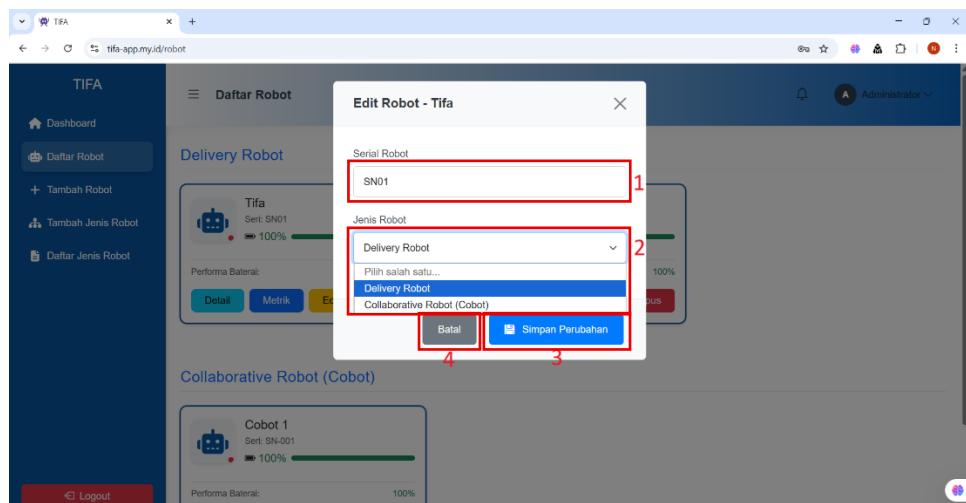
Setelah klik tombol “Metrik” *user* akan diarahkan menuju halaman metrik yang menampilkan informasi mengenai robot yang divisualisasikan menggunakan grafik. Terdapat 3 grafik yaitu grafik aktivitas robot, banyaknya pesanan per hari, dan performa baterai.



**Gambar 4. 38 Halaman Metrik**

## F. Edit

Setelah klik tombol “Edit” akan muncul *pop-up* edit robot untuk edit nomor seri robot dan jenis robot.

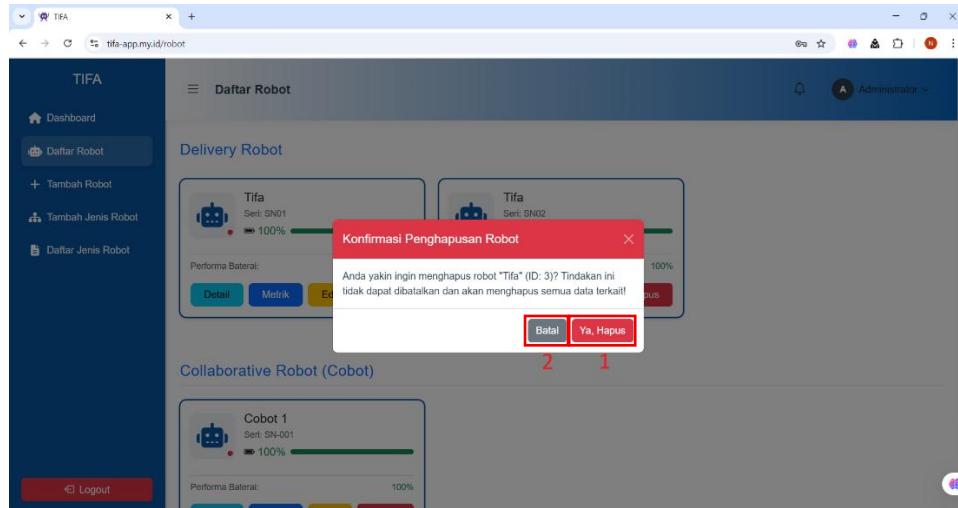


**Gambar 4. 39 Pop-up Edit Robot**

1. Masukkan nomor seri baru, apabila ingin merubah nomor seri robot.
2. Klik *dropdown* untuk mengganti jenis robot. Pada *dropdown* terdapat pilihan jenis robot yang sudah terdaftar, *user* dapat memilih salah satu jenis robot.
3. Apabila sudah selesai melakukan edit nomor seri robot dan/atau jenis robot klik tombol “Simpan Perubahan” untuk menyimpan perubahan.
4. Klik tombol “Batal” apabila tidak ingin melakukan perubahan apapun.

## G. Hapus

Setelah klik tombol “Hapus” akan muncul *pop-up* konfirmasi penghapusan robot.

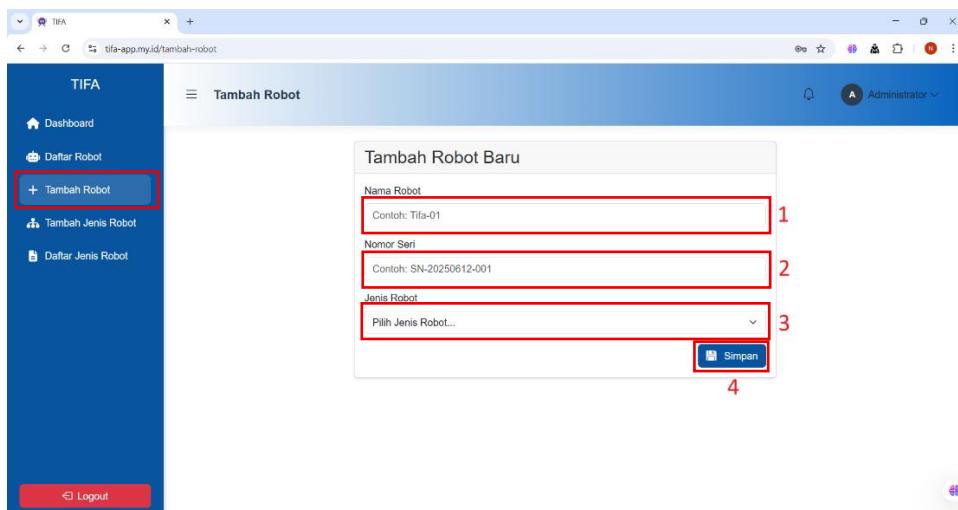


**Gambar 4. 40 Pop-up Konfirmasi Penghapusan Robot**

1. Klik tombol “Ya, Hapus” apabila ingin menghapus robot beserta dengan semua data yang terkait.
2. Klik tombol “Batal” apabila ingin membatalkan tindakan penghapusan robot.

## H. Tambah Robot

Untuk menambah robot baru pilih menu “Tambah Robot” pada *sidebar*. Halaman tambah robot menampilkan *form* tambah robot baru yang dapat diisi oleh *user*.



**Gambar 4. 41 Halaman Tambah Robot**

1. Masukkan nama robot.
2. Masukkan nomor seri robot.

3. Pilih jenis robot pada *dropdown*.
4. Klik tombol “Simpan” untuk menambah robot baru. Setelah klik tombol “Simpan” robot akan bertambah dan ditampilkan pada halaman daftar robot.

### I. Tambah Jenis Robot

Untuk menambah jenis robot baru pilih menu “Tambah Jenis Robot” pada *sidebar*. Halaman tambah jenis robot menampilkan *form* tambah jenis robot baru yang dapat diisi oleh *user*.



**Gambar 4. 42 Halaman Tambah Jenis Robot**

1. Masukkan nama jenis robot.
2. *Checklist checkbox* “Jenis ini memiliki fitur Peta (*Maps*)” apabila jenis robot baru yang akan ditambahkan memiliki fitur navigasi.
3. Klik tombol “Simpan” untuk menambah jenis robot baru. Setelah klik tombol “Simpan” jenis robot akan bertambah dan ditampilkan pada halaman daftar jenis robot dan *dropdown* pilih jenis robot.

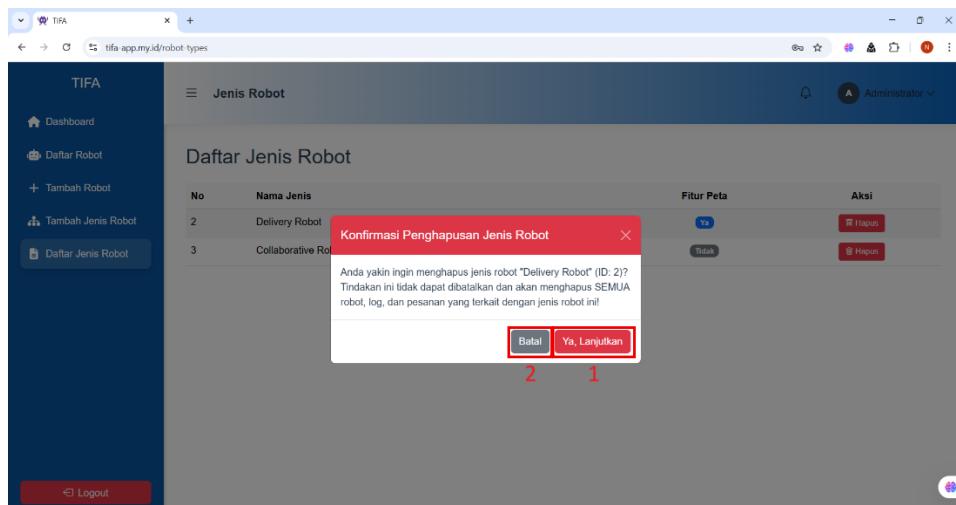
### J. Daftar Jenis Robot

Pilih menu “Daftar Jenis Robot” pada *sidebar* untuk melihat semua jenis robot yang terdaftar.

No	Nama Jenis	Fitur Peta	Aksi
2	Delivery Robot	Ya	Hapus
3	Collaborative Robot (Cobot)	Tidak	Hapus

**Gambar 4. 43 Halaman Daftar Jenis Robot**

Klik tombol “Hapus” apabila ingin menghapus jenis robot beserta robot dengan jenis tersebut dan semua data yang terkait. Apabila klik tombol “Hapus” akan muncul *pop-up* konfirmasi penghapusan jenis robot.

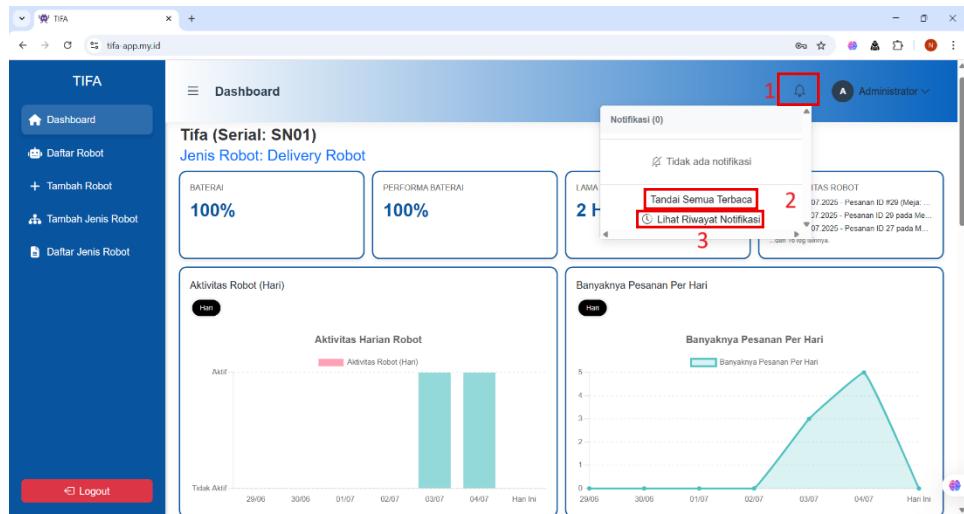


**Gambar 4. 44 Pop-up Konfirmasi Penghapusan Jenis Robot**

1. Klik tombol “Ya, Hapus” apabila ingin menghapus jenis robot beserta robot dengan jenis tersebut dan semua data yang terkait.
2. Klik tombol “Batal” apabila ingin membatalkan tindakan penghapusan jenis robot.

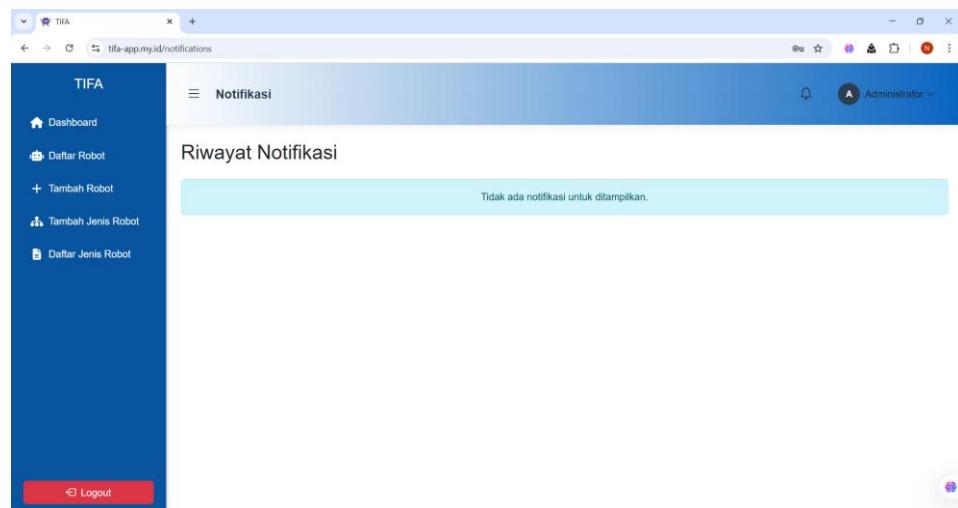
## K. Notifikasi

Apabila ingin melihat notifikasi peringatan terkait robot, *user* dapat klik ikon lonceng. Notifikasi peringatan yang ditampilkan yaitu notifikasi baterai robot lemah dan performa baterai robot menurun disertai dengan saran tindakan yang perlu dilakukan.



**Gambar 4. 45 Pop-up Notifikasi**

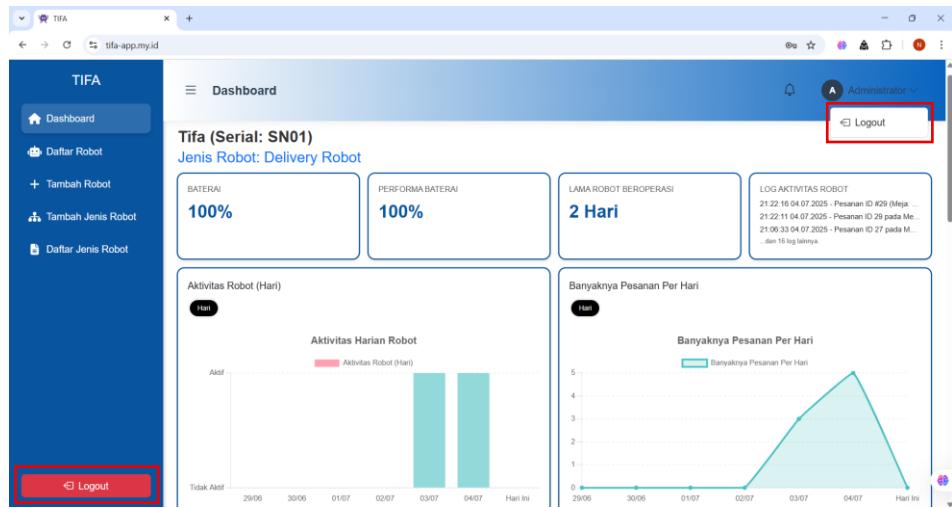
1. Klik ikon lonceng untuk membuka *pop-up* notifikasi.
2. Klik tombol “Tandai Semua Terbaca” untuk mengubah status notifikasi menjadi telah terbaca.
3. Klik tombol “Lihat Riwayat Notifikasi” dan *user* akan diarahkan ke halaman riwayat notifikasi yang menampilkan seluruh *history* notifikasi.



**Gambar 4. 46 Halaman Riwayat Notifikasi**

## L. Logout

Apabila *user* ingin mengakhiri sesi monitoring, *user* dapat klik tombol “Logout” pada sidebar atau pada *user account dropdown*. Setelah klik tombol “Logout”, *user* akan diarahkan ke halaman *login*.



**Gambar 4. 47 Tombol Logout**

#### 4.3.3 Penyerahan Produk kepada Mitra

Penyerahan produk berupa aplikasi *mobile* dan web monitoring sebagai sistem kontrol dan monitoring robot TIFA kepada mitra terkait Tel-U Coffee.



**Gambar 4. 48 Penyerahan Produk kepada Mitra**

Produk sistem Monitoring Tel-U Interactive Food Assistant (TIFA) hasil pengembangan dari sistem sebelumnya berhasil memenuhi kebutuhan operasional di Tel-U Coffee dengan menyediakan solusi monitoring dan pengendalian robot pengantar makanan secara *real-time* [10]. Produk ini meliputi aplikasi *mobile* dan web monitoring yang terintegrasi dengan perangkat keras robot, sehingga memudahkan staf dalam mengatur pengantaran pesanan, memantau status robot, serta meningkatkan efisiensi pelayanan terutama pada jam sibuk.

## BAB 5

# PENGUJIAN

### 5.1 Skenario Umum Pengujian

Sebelum pelaksanaan pengujian sistem TIFA (Tel-U Interactive Food Assistant), penyusunan skenario pengujian yang komprehensif merupakan langkah kritis yang dilakukan tim pengembang. Skenario ini berfungsi sebagai panduan operasional untuk memverifikasi bahwa setiap komponen sistem mulai dari aplikasi dan web monitoring berfungsi dengan baik. Tanpa skenario yang terdefinisi dengan baik, risiko *bug* tak terdeteksi dan ketidaksesuaian dengan kebutuhan pengguna meningkat signifikan. Sebaliknya, pendekatan terstruktur memungkinkan identifikasi dini masalah desain, optimalisasi stabilitas sistem, dan validasi akhir bahwa solusi TIFA benar-benar menjawab permasalahan sistem robot operasional di Tel-U Coffee yang dianalisis dalam CD-1 penanganan lonjakan pesanan pada saat jam sibuk, peningkatan kepuasan pelanggan, peningkatan aplikasi dan pembuatan web monitoring [54]. Berdasarkan kompleksitas arsitektur TIFA yang melibatkan aplikasi android dan web monitoring, pengujian dijalankan secara bertahap mengikuti alur sebagai berikut.

Sebelum pelaksanaan pengujian sistem TIFA (Tel-U Interactive Food Assistant), penyusunan skenario pengujian yang komprehensif merupakan langkah kritis yang dilakukan tim pengembang. Skenario ini berfungsi sebagai panduan operasional untuk memverifikasi bahwa setiap komponen sistem mulai dari aplikasi dan web monitoring berfungsi dengan baik. Tanpa skenario yang terdefinisi dengan baik, risiko *bug* tak terdeteksi dan ketidaksesuaian dengan kebutuhan pengguna meningkat signifikan. Sebaliknya, pendekatan terstruktur memungkinkan identifikasi dini masalah desain, optimalisasi stabilitas sistem, dan validasi akhir bahwa solusi TIFA benar-benar menjawab permasalahan sistem robot operasional di Tel-U Coffee yang dianalisis dalam CD-1 penanganan lonjakan pesanan pada saat jam sibuk, peningkatan kepuasan pelanggan, peningkatan aplikasi dan pembuatan web monitoring [54]. Berdasarkan kompleksitas arsitektur TIFA yang melibatkan aplikasi android dan web monitoring, pengujian dijalankan secara bertahap mengikuti alur sebagai berikut.

#### 5.1.1 Security Testing

Pengujian ini difokuskan pada aspek keamanan sistem untuk memastikan bahwa web monitoring bebas dari celah keamanan yang dapat dieksloitasi oleh pihak tidak bertanggung jawab. Pengujian mencakup pengecekan header keamanan, pengungkapan metadata, serta pemeriksaan komentar mencurigakan pada kode sumber. Proses pengujian dilakukan secara

otomatis menggunakan alat bantu OWASP ZAP yang mampu mendeteksi berbagai kerentanan umum dalam web, baik melalui pemindaian pasif maupun aktif.

### **5.1.2 *Integration Testing***

Pengujian difokuskan pada keterpaduan antar komponen sistem, seperti komunikasi antara aplikasi, web, dan robot, sinkronisasi sensor IR pada *tray*, kesesuaian koordinat meja yang ditampilkan pada seluruh antarmuka aplikasi, serta kesesuaian seluruh data informasi robot yang ditampilkan pada seluruh antarmuka web monitoring. Hal ini penting untuk memastikan aliran data antar bagian sistem berlangsung dengan stabil dan akurat [55].

### **5.1.3 *Alpha Testing***

Dilakukan oleh tim pengembang dengan pendekatan *black-box Testing* untuk menguji fungsi-fungsi utama sistem robot, seperti *respons* terhadap perintah, pendekeksian *tray* makanan, dan keterbacaan data informasi robot. Pengujian ini bertujuan memastikan fungsionalitas sistem berjalan sebagaimana mestinya sebelum diuji oleh pengguna eksternal [56].

### **5.1.4 *User Acceptance Testing (UAT)***

Dilakukan sebagai validasi akhir terhadap sistem oleh pihak pengguna langsung. Uji ini mencakup simulasi operasional selama beberapa hari serta survei kepuasan pengguna untuk menilai apakah sistem telah memenuhi kebutuhan dan layak digunakan secara resmi dalam lingkungan kerja sehari-hari [57], [58].

Dilakukan sebagai validasi akhir terhadap sistem oleh pihak pengguna langsung. Uji ini mencakup simulasi operasional selama beberapa hari serta survei kepuasan pengguna untuk menilai apakah sistem telah memenuhi kebutuhan dan layak digunakan secara resmi dalam lingkungan kerja sehari-hari [57], [58].

## **5.2 Detil Pengujian**

Pada bagian ini disajikan skenario pengujian yang dirancang untuk mengevaluasi performa dan keandalan sistem robot TIFA (Tel-U Interactive Food Assistant) berdasarkan masing-masing jenis pengujian. Setiap skenario disusun untuk mencerminkan kondisi penggunaan nyata, baik dari sisi teknis sistem maupun interaksi pengguna. Selain itu, hasil pengujian juga disertakan guna memberikan gambaran terhadap efektivitas implementasi sistem dalam menjawab kebutuhan operasional. Melalui pemaparan skenario dan hasil

pengujian ini, dapat dinilai sejauh mana sistem memenuhi kriteria fungsionalitas, stabilitas, serta penerimaan oleh pengguna akhir.

### **5.2.1 Security Testing**

Sebagai bagian dari proses validasi sistem, pengujian keamanan (*Security Testing*) dilakukan untuk memastikan bahwa aplikasi yang dikembangkan memiliki ketahanan terhadap berbagai potensi serangan siber. Pengujian ini penting guna mengidentifikasi celah-celah keamanan yang dapat membahayakan integritas data, kerahasiaan informasi, maupun ketersediaan layanan dalam aplikasi. Dalam subbab ini, disajikan skenario pengujian secara rinci, proses teknis yang ditempuh selama pengujian, serta hasil yang diperoleh dari aktivitas pengujian tersebut.

#### **A. Skenario Detil**

Pengujian keamanan dilakukan terhadap aplikasi Juice Shop yang dijalankan pada alamat <http://tifa-app.my.id>. Fokus utama dari skenario ini adalah untuk mengidentifikasi potensi kerentanan yang dapat dieksplorasi oleh pihak tidak bertanggung jawab, termasuk kebocoran metadata *cloud*, kekurangan konfigurasi header keamanan, dan potensi pengungkapan informasi sensitif dari kode sumber. Skenario pengujian mencakup simulasi serangan aktif dan pasif untuk menilai bagaimana aplikasi merespons terhadap berbagai jenis risiko keamanan.

#### **B. Proses Pengujian**

Pengujian dilakukan menggunakan alat OWASP ZAP versi 2.15.0, dengan teknik pemindaian aktif dan pasif. Semua konteks dan situs yang tersedia disertakan tanpa penyaringan. Pemindaian dilakukan terhadap berbagai *endpoint*, termasuk halaman *login*, robots.txt, sitemap.xml, dan file JavaScript internal (main.js). ZAP secara otomatis mengidentifikasi jenis header HTTP yang hilang, metadata *cloud* yang terekspos, dan komentar mencurigakan di dalam file sumber. Risiko dinilai berdasarkan kombinasi antara tingkat keparahan (Tinggi, Menengah, Rendah, Informasi) atas hasil deteksi.

#### **C. Hasil Pengujian**

Setelah seluruh proses pengujian keamanan dilakukan menggunakan alat OWASP ZAP, tahap ini menyajikan hasil temuan yang berhasil diidentifikasi selama pengujian. Setiap kerentanan yang ditemukan dicatat dan diklasifikasikan berdasarkan tingkat risikonya, serta disertai informasi mengenai *endpoint* yang terdampak dan keterangan deskriptif. Penyajian hasil ini bertujuan untuk memberikan gambaran yang jelas mengenai kondisi keamanan

aplikasi, serta menjadi dasar pertimbangan dalam melakukan perbaikan dan penguatan sistem di tahap pengembangan selanjutnya.

**Tabel 5. 1 Hasil Pengujian Security Testing Web Monitoring**

No.	Jenis kerentanan	Risiko	Endpoint Terdeteksi	Keterangan
1.	<i>Cloud Metadata Potentially Exposed</i>	Tinggi	/latest/meta-data/?redirect=/	Metadata <i>cloud</i> dapat diakses publik
2.	<i>Content Security Policy (CSP) Header Not Set</i>	Menengah	/login? redirect=/	Tidak ada CSP, berisiko terhadap serangan XSS
3.	<i>Missing Anti-clickjacking Header</i>	Menengah	/robots.txt	Header X-Frame-Options tidak diatur
4.	<i>X-Content-Type-Options Header Missing</i>	Rendah	/robots.txt	Header untuk mencegah sniffing MIME hilang
5.	<i>Modern Web Application</i>	Informasi	/sitemap.xml	Deteksi aplikasi modern (informasi umum)
6.	<i>Information Disclosure- Suspicious Comments</i>	Informasi	/src/main.js	Komentar mencurigakan yang dapat mengungkap informasi sensitif

### 5.2.2 *Integration Testing*

*Integration Testing* berfokus pada pengujian integrasi antara aplikasi, web, dan database. Tujuannya adalah untuk memastikan bahwa aplikasi android dan web monitoring dapat berkomunikasi dengan database dengan benar dan mendapatkan data yang diharapkan.

#### A. Skenario Detil Aplikasi

Tabel ini merinci skenario pengujian integrasi untuk aplikasi TIFA (Tel-U Interactive Food Assistant), termasuk tindakan yang diuji, *endpoint API* yang terlibat, dan output yang diharapkan.

**Tabel 5. 2 Skenario Detil *Integration Testing* Aplikasi**

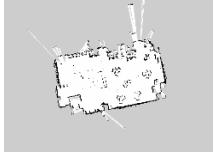
No.	Skenario	TIFA API Endpoint	Output yang Diharapkan
1.	Mengambil data peta yang tersimpan	<a href="https://be.tifa-app.my.id/api/images">https://be.tifa-app.my.id/api/images</a>	Semua file blob yang tersimpan dapat ditampilkan
2.	Mengambil semua nomor meja		Dapat menampilkan semua nomor meja yang tersimpan
3.	Menambahkan nomor meja ke database		Nomor meja berhasil ditambahkan dan ditampilkan di ui
4.	Menghapus nomor meja dari database	<a href="https://be.tifa-app.my.id/api/app/coordinates">https://be.tifa-app.my.id/api/app/coordinates</a>	Nomor meja berhasil dihapus dari database dan ui
5.	Mengambil koordinat yang sesuai dari nomor meja yang dipilih		Koordinat dapat diteruskan dan dipakai untuk nomor meja
6.	Menampilkan kondisi baterai dari database	<a href="https://be.tifa-app.my.id/api/app/battery_level">https://be.tifa-app.my.id/api/app/battery_level</a>	Baterai menampilkan persentase terbaru
7.	Memasukkan nomor meja dan posisi tray ke database	<a href="https://be.tifa-app.my.id/api/orders">https://be.tifa-app.my.id/api/orders</a>	Nomor meja dan posisi tray berhasil disimpan ke database
8.	Mengambil nomor meja dan posisi tray dari database	<a href="https://be.tifa-app.my.id/api/app/latest-orders">https://be.tifa-app.my.id/api/app/latest-orders</a>	Nomor meja dan posisi tray berhasil diambil dari database
9.	Menghapus nomor meja dan posisi tray dari database	<a href="https://be.tifa-app.my.id/api/app/delete-order">https://be.tifa-app.my.id/api/app/delete-order</a>	Nomor meja dan posisi tray berhasil dihapus dari database
10.	Mengubah kondisi pintu	<a href="https://be.tifa-app.my.id/api/app/door-status">https://be.tifa-app.my.id/api/app/door-status</a>	Pintu dapat berubah tertutup dan terbuka

No.	Skenario	TIFA API Endpoint	Output yang Diharapkan
11.	Menerima perubahan status	<a href="https://be.tifa-app.my.id/api/app/update-order-status">https://be.tifa-app.my.id/api/app/update-order-status</a>	Status pengantaran akan diperbarui di database
12.	Mengirim data pesanan ke robot	<a href="https://be.tifa-app.my.id/api/app/most-recent-orders">https://be.tifa-app.my.id/api/app/most-recent-orders</a>	Robot dapat menerima data pesanan

## B. Proses Pengujian Aplikasi

Tabel ini menjelaskan proses data yang diambil untuk melakukan pengujian integrasi pada aplikasi TIFA.

**Tabel 5. 3 Proses *Integration Testing* Aplikasi**

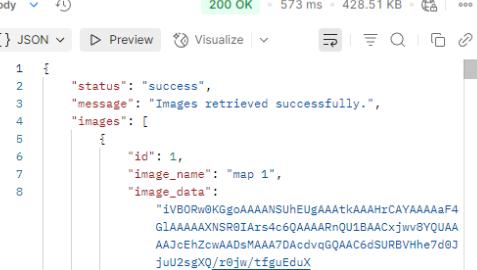
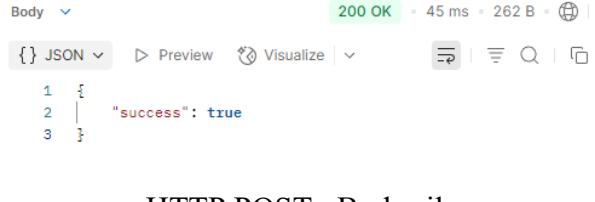
No.	Skenario	Tipe Data	Value
1.	Mengambil data peta yang tersimpan	Blob	
2.	Mengambil semua nomor meja	Json	{"id":1,"table_number": "Meja 1", "coordinates": 1}
3.	Menambahkan nomor meja ke database	String	"table_number": "Meja 1", "coordinates": "1"
4.	Menghapus nomor meja dari database	Json	{"id":1,"table_number": "Meja 1", "coordinates": 1}
5.	Mengambil koordinat yang sesuai dari nomor meja yang dipilih	String	"coordinates": 1
6.	Menampilkan kondisi baterai dari database	String	"battery_level":100
7.	Memasukkan nomor meja dan posisi <i>tray</i> ke database	Json	{"order": 123, "tray": "Atas", "table": "Meja 5"}

No.	Skenario	TIFA API <i>Endpoint</i>	<i>Output</i> yang Diharapkan
8.	Mengambil nomor meja dan posisi <i>tray</i> dari database	Json	{"order_number": 123, "tray_position": "Atas", "table_number": "Meja 1", "status": 0}
9.	Menghapus nomor meja dan posisi <i>tray</i> dari database	Json	{"order_number": 123, "tray_position": "Atas", "table_number": "Meja 1", "status": 0}
10.	Mengubah kondisi pintu	Int	"Command":1
11.	Mengirim data pesanan ke robot	Json	{"order_number": 123, "tray_position": "Atas", "table_number": "Meja 5", "coordinates": 100, "created_at": "2025-06-14 20:25:00", "status": 0}
12.	Menerima data dari robot	Json	{"order_number": 123, "tray_position": "Atas", "table_number": "Meja 5", "coordinates": 100, "created_at": "2025-06-14 20:25:00", "status": 1}

### C. Hasil Pengujian Aplikasi

Tabel ini menyajikan hasil dari pengujian integrasi yang dilakukan pada aplikasi TIFA berfungsi secara terpadu dan sesuai dengan spesifikasi yang telah ditetapkan.

**Tabel 5. 4 Hasil *Integration Testing* Aplikasi**

No.	Skenario	TIFA API <i>Endpoint</i>	Output yang Diharapkan
1.	Mengambil data peta yang tersimpan	https://be.tifa-app.my.id/api/images	<p>HTTP GET : Berhasil</p> 
2.	Mengambil semua nomor meja		<p>HTTP GET : Berhasil</p> 
3.	Menambahkan nomor meja ke database	https://be.tifa-app.my.id/api/app/coordinates?table_number	<p>HTTP POST : Berhasil</p> 
4.	Menghapus nomor meja dari database		<p>HTTP POST : Berhasil</p> 
5.	Mengambil koordinat yang sesuai dari nomor meja yang dipilih		<p>HTTP GET : Berhasil</p> 

No.	Skenario	TIFA API Endpoint	Output yang Diharapkan
6.	Menampilkan kondisi baterai dari database	https://be.tifa-app.my.id/api/app/battery-level/3	<p>HTTP GET : Berhasil</p> <pre> 1 { 2   "status": "success", 3   "message": "Level baterai berhasil diambil.", 4   "success": true, 5   "battery_level": 100, 6   "timestamp": "2025-07-03 13:11:05" 7 } </pre>
7.	Memasukkan nomor meja dan posisi tray ke database	https://be.tifa-app.my.id/api/orders	<p>HTTP POST : Berhasil</p> <pre> 1 { 2   "status": "success", 3   "message": "Pesanan berhasil dibuat.", 4   "data": { 5     "order": { 6       "id": 30, 7       "robot_id": 3, 8       "order_number": 1, 9       "tray_position": "Atas", 10      "table_number": "Meja 1", 11      "coordinates": "[6.0,1.0]", 12      "status": 0, 13      "created_at": "2025-07-06 06:15:48" 14    } 15  } 16} </pre>
8.	Mengambil nomor meja dan posisi tray dari database	https://be.tifa-app.my.id/api/app/latest-orders	<p>HTTP GET : Berhasil</p> <pre> 1 { 2   "status": "success", 3   "message": "Data pesanan terbaru berhasil diambil.", 4   "orders": [ 5     { 6       "order_number": 1, 7       "tray_position": "Atas", 8       "table_number": "Meja 1", 9       "status": 0 10    } 11  ] 12} </pre>
9.	Menghapus nomor meja dan posisi tray dari database	https://be.tifa-app.my.id/api/app/delete-order	<p>HTTP POST : Berhasil</p> <pre> 1 { 2   "success": true 3 } </pre>
10.	Mengubah kondisi pintu	https://be.tifa-app.my.id/api/app/door-status	<p>HTTP POST : Berhasil</p> <pre> 1 { 2   "status": "success", 3   "message": "Status pintu berhasil diambil.", 4   "door_status": 1 5 } </pre>

No.	Skenario	TIFA API Endpoint	Output yang Diharapkan
11.	Mengirim data pesanan ke robot	https://be.tifa-app.my.id/api/app/most-recent-orders	<p>Body  200 OK • 149 ms • 769 B        </p> <pre> 1  { 2    "status": "success", 3    "message": "Data pesanan paling baru berhasil diambil.", 4    "orders": [ 5      { 6        "id": 30, 7        "robot_id": 3, 8        "order_number": 1, 9        "tray_position": "Atas", 10       "table_number": "Meja 1", 11       "coordinates": "[6.0,1.0]", 12       "status": 0, 13       "created_at": "2025-07-06 06:15:48" </pre> <p style="text-align: center;">HTTP GET : Berhasil</p>
12.	Menerima data dari robot	https://be.tifa-app.my.id/api/app/update-order-status	<p>Body  200 OK • 533 ms • 759 B          </p> <pre> 1  { 2    "status": "success", 3    "message": "Status pesanan berhasil diperbarui.", 4    "order": { 5      "id": 30, 6      "robot_id": 3, 7      "order_number": 1, 8      "tray_position": "Atas", 9      "table_number": "Meja 1", 10     "coordinates": "[6.0,1.0]", 11     "status": 1, 12     "created_at": "2025-07-06 06:15:48" </pre> <p style="text-align: center;">HTTP GET : Berhasil</p>

#### D. Skenario Detil Web Monitoring

Tabel ini merinci skenario pengujian integrasi untuk web monitoring, termasuk tindakan yang diuji, *endpoint* API yang terlibat, dan *output* yang diharapkan.

**Tabel 5. 5 Skenario Detil *Integration Testing* Web Monitoring**

No.	Skenario	Web API Endpoint	Output yang Diharapkan
1.	Mengambil semua data robot	/api/robotData	Berhasil menampilkan seluruh data robot
2.	Melakukan <i>login</i>	/api/login	<i>Login</i> berhasil
3.	Menambahkan data robot	/api/robotData	Data robot baru berhasil ditambahkan dan dikonfirmasi dengan <i>response</i> sukses

No.	Skenario	Web API <i>Endpoint</i>	<i>Output</i> yang Diharapkan
4.	Menambahkan data jenis robot	/api/robotType	Jenis robot baru berhasil ditambahkan
5.	Mengambil data peta yang tersimpan	/api/images	Semua file blob yang tersimpan dapat ditampilkan
6.	Mengambil data jenis robot	/api/robotType	Daftar jenis robot berhasil ditampilkan
7.	Mengambil data Detail robot	/api/robotData /:robotId	Detail informasi robot berhasil ditampilkan berdasarkan id
8.	Melakukan <i>edit</i> data robot	/api/robotData /:robotId	Data robot berhasil diperbarui dan <i>respons</i> sukses diterima

## E. Proses Pengujian Web Monitoring

Tabel ini menjelaskan proses data yang diambil untuk melakukan pengujian integrasi pada web monitoring.

**Tabel 5. 6 Proses *Integration Testing* Web Monitoring**

No.	Skenario	Tipe Data	Value
1.	Mengambil semua data robot	Json	{"robot_type": {"id": 1, "nama": "Robot Pengantar Makanan"}, "battery_data": [{"id": 1, "name": "TIFA", "serial": "SN-001", "battery_performace": 100, "battery_level": 100, "timestamp": "2025-06-19 07:58:38", "created_at": "2025-06-19 07:58:38", "updated_at": "2025-06-19 21:21:18", "jenis_robot_id": 1}}

No.	Skenario	Tipe Data	Value
2.	Melakukan <i>login</i>	Json	{"message": "Login berhasil.", "token_type": "Bearer", "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9eyJpc3MiOiJodHRwOi8vMTI3LjAuMC4xOjgwMDAiLCJhdWIQiOjodHRwOi8vMTI3LjAuMC4xOjgwMDAiLCJpYXQiOjE3NTAzMzk4NTgsIm5iZiI6MTc1MDMzOTg1OCwiZXhwIjoxNzUwMzQzNDU4LCJkYXRhIjp7InVzZXJJZCI6MSwidXNlcm5hbWUiOiJhZG1pbilIsIm5hbWUiOiJBZG1pbmlzdHJhdG9yIn19.vIHxRqVD27Q9nICSV3Wm7lFIxoqggvWjUqAvbGdJGX", "expires_in": 3600, "user": {"id": 1, "username": "admin", "name": "Administrator"}}
3.	Menambahkan data robot	Json	{"message": "Robot/battery data successfully saved.", "battery_data_entry": {"id": 2, "name": "TIFA", "serial": "SN-001", "battery_level": 100, "battery_performace": 100, "timestamp": "2025-06-19 21:04:26", "created_at": "2025-06-19 21:04:26", "updated_at": "2025-06-19 21:04:26"}}
4.	Menambahkan data jenis robot	Json	{"message": "Robot type successfully added.", "robot_type": {"id": 2, "nama": "Autonomous Mobile Robot (AMR)"}}}
5.	Mengambil data peta yang tersimpan	Blob	
6.	Mengambil data jenis robot	Json	{"message": "Robot types retrieved successfully.", "robot_types": [{"id": 1, "nama": "Robot Pengantar Makanan"}, {"id": 2, "nama": "Autonomous Mobile Robot (AMR)"}]}

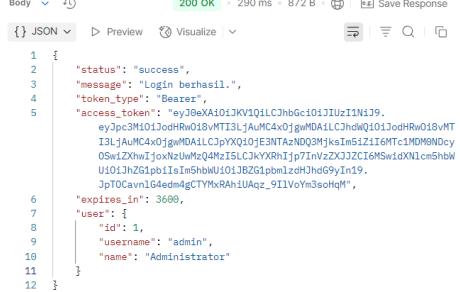
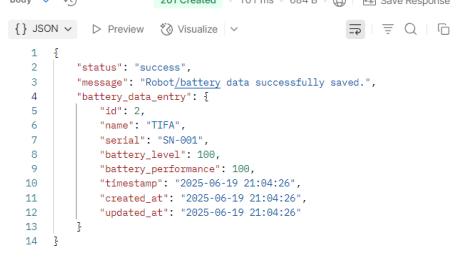
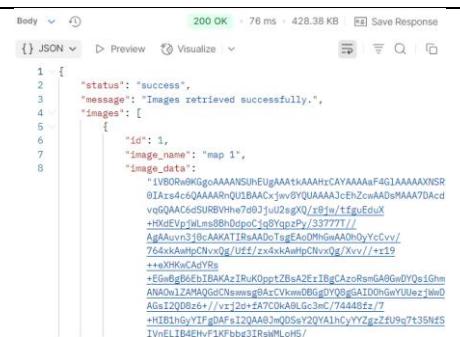
No.	Skenario	Tipe Data	Value
7.	Mengambil data Detail robot	Json	{"message": "Battery data grouped by robot types retrieved successfully.", "data": [{"robot_type": {"id": 1, "nama": "Robot Pengantar Makanan"}, "battery_data": [{"id": 1, "name": "TIFA", "serial": "SN-001", "battery_performace": 100, "battery_level": 100, "timestamp": "2025-06-19 07:58:38", "created_at": "2025-06-19 07:58:38", "updated_at": "2025-06-19 21:21:18", "jenis_robot_id": 1}]}]
8.	Melakukan edit data robot	Json	{"message": "Data robot ID 1 berhasil diperbarui.", "robot_data": {"id": 1, "name": "TIFA", "serial": "SN-001", "battery_level": 20, "battery_performace": 99, "timestamp": "2025-06-19 07:58:38", "created_at": "2025-06-19 07:58:38", "updated_at": "2025-06-19 21:57:09"}}

## F. Hasil Pengujian Web Monitoring

Tabel ini menyajikan hasil dari pengujian integrasi yang dilakukan pada web monitoring.

**Tabel 5. 7 Hasil Integration Testing Web Monitoring**

No.	Skenario	Web API Endpoint	Hasil
1.	Mengambil semua data robot	/api/robotData	 HTTP GET : Berhasil

No.	Skenario	Web API Endpoint	Hasil
2.	Melakukan <i>login</i>	/api/login	<p>HTTP POST : Berhasil</p>  <pre> 1 { 2     "status": "success", 3     "message": "Login berhasil.", 4     "token_type": "Bearer", 5     "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9. 6         eyJpZ3M10jodRw0i6WTT3LjAuMC4xJgMDA1LC1pYXQ10jE3NTAxNDQ3MjksIm5iZlIiMTc1MDM0NDcy 7         OSwiZXhwIjoxNzUwMzQ4MzLSLjklXXRhIjp7InVzZXJ2C16MSwldXm1cm5hbWl 8         U101JhZG1pb1sIm5hNU101JBZG1pbm1nzdJh0g9yIn19. 9         JpTOCavnLG4ednAGCTYmRAHuiAqz_911VoVmSsohqt", 10    "expires_in": 3600, 11    "user": { 12        "id": 1, 13        "username": "admin", 14        "name": "Administrator" 15    } 16 } </pre>
3.	Menambahkan data robot	/api/robotData	<p>HTTP POST : Berhasil</p>  <pre> 1 { 2     "status": "success", 3     "message": "Robot/battery data successfully saved.", 4     "battery_data_entry": { 5         "id": 2, 6         "name": "TIFA", 7         "serial": "SN-001", 8         "battery_level": 100, 9         "battery_performance": 100, 10        "timestamp": "2025-06-19 21:04:26", 11        "created_at": "2025-06-19 21:04:26", 12        "updated_at": "2025-06-19 21:04:26" 13    } 14 } </pre>
4.	Menambahkan data jenis robot	/api/robotType	<p>HTTP POST : Berhasil</p>  <pre> 1 { 2     "status": "success", 3     "message": "Robot type successfully added.", 4     "robot_type": { 5         "id": 2, 6         "nama": "Autonomous Mobile Robot (AMR)" 7     } 8 } </pre>
5.	Mengambil data peta yang tersimpan	/api/images	<p>HTTP GET : Berhasil</p>  <pre> 1 { 2     "status": "success", 3     "message": "Images retrieved successfully.", 4     "images": [ 5         { 6             "id": 1, 7             "image_name": "map 1", 8             "image_data": "iVBORw0KGgoAAAANSUhEUgAAAtkAAAHrCAYAAaF4G1AAAAAXNSR 9             0IArs4c6QAAAReQ1U5BAcXjwBYQAAA3jEhZcaADsMAAA7DAc 10            vqfQAAc6cSUrVhne7d0jul2gxQ/\v9i/\vfg/EduX 11            +HxDfUpVlns88hDgcCqg8YzpP/33777//AgAvun310cAAKAT1B 12            sAd0TsgeA0o9hgAA0h0yYcCvv/764xxAmhpCNvx0g/Uff/zx4kk 13            wmpCNvx0g/Xvv/+19++XHkwAd/R8+EGb6EBIBKAz1RuK0pptZ 14            BsA2Ez1BgCazoRsmG49Gw0YQs1Gh 15            ANAw0iZAM4QgCmNmmsgAarCVkm06GgDQYg@A100hGwYUu 16            zjWw0Ag512Q08z67//vz12e+fA7C0kaABLGc3mC74446z7/7 17            H1B1NgYTFgAfAf12QAABj0OsY20YAlhCyyY2gzFu9q7t3N4s 18            IVnEL1B4EHvF1KFbde3IRs@MLcH5/" 19 } 20 ] </pre>

No.	Skenario	Web API Endpoint	Hasil
6.	Mengambil data jenis robot	/api/robotType	<p>HTTP GET : Berhasil</p> <pre> 1 { 2   "status": "success", 3   "message": "Robot types retrieved successfully.", 4   "robot_types": [ 5     { 6       "id": 1, 7       "nama": "Robot Pengantar Makanan" 8     }, 9     { 10       "id": 2, 11       "nama": "Autonomous Mobile Robot (AMR)" 12     } 13   ] 14 } </pre>
7.	Mengambil data Detail robot	/api/robotData/:robotId	<p>HTTP GET : Berhasil</p> <pre> 1 { 2   "status": "success", 3   "message": "Battery data grouped by robot types retrieved successfully.", 4   "data": [ 5     { 6       "robot_type": { 7         "id": 1, 8         "name": "Robot Pengantar Makanan" 9       }, 10      "battery_data": [ 11        { 12          "id": 1, 13          "name": "TIFA", 14          "serial": "SN-001", 15          "battery_performance": 100, 16          "battery_level": 100, 17          "timestamp": "2025-06-19 07:58:38", 18          "created_at": "2025-06-19 07:58:38", 19          "updated_at": "2025-06-19 21:21:18", 20          "jenis_robot_id": 1 21        }, 22        { 23          "id": 2, 24          "name": "TIFA", 25          "serial": "SN-002", 26          "battery_performance": 100, 27        } 28      ] 29    } 30  ] 31 } </pre>
8.	Melakukan edit data robot	/api/robotData/:robotId	<p>HTTP PUT : Berhasil</p> <pre> 1 { 2   "status": "success", 3   "message": "Data robot ID 1 berhasil diperbarui.", 4   "robot_data": [ 5     { 6       "id": 1, 7       "name": "TIFA", 8       "serial": "SN-001", 9       "battery_level": 29, 10      "battery_performance": 99, 11      "timestamp": "2025-06-19 07:58:38", 12      "created_at": "2025-06-19 07:58:38", 13      "updated_at": "2025-06-19 21:57:09" 14    } 15  ] 16 } </pre>

## G. Skenario Detil IR Sensor

Pada tahap ini, dilakukan pengujian fungsional terhadap sistem sensor infrared (IR) yang diterapkan pada *tray* robot Tel-U Interactive Food Assistant (TIFA). Tujuan utama dari pengujian ini adalah untuk memastikan bahwa IR sensor dapat secara akurat mendeteksi keberadaan makanan/minuman pada setiap *tray* dalam kondisi operasional di Tel-U Coffee.

Pengujian sensor infrared (IR) dilakukan dengan cakupan skenario yang dirancang menyerupai kondisi operasional nyata di lingkungan kafe. Lima buah sensor IR dipasang pada masing-masing posisi *tray* pada robot pengantar makanan. Pengujian dilaksanakan di dalam ruangan terang dengan pencahayaan aktif, setara dengan intensitas cahaya  $\pm 500\text{--}800$  lux

sebagaimana kondisi normal di area kafe. Dalam skenario ini, seluruh *tray* dibiarkan dalam keadaan kosong guna memvalidasi apakah sensor dapat mengenali ketiadaan objek secara akurat. Tujuan khusus dari pengujian ini adalah untuk memastikan bahwa seluruh sensor IR memberikan *output* logika LOW saat tidak ada makanan/minuman pada *tray*, sehingga sistem dapat menutup pintu secara otomatis dan mengaktifkan tombol konfirmasi pada antarmuka pengguna untuk melanjutkan proses pengiriman berikutnya.

## H. Proses Pengujian IR Sensor

Proses pengujian sensor infrared (IR) dilakukan untuk memastikan akurasi deteksi keberadaan objek (makanan/minuman) pada setiap *tray* robot Tel-U Interactive Food Assistant (TIFA) dalam kondisi pencahayaan normal. Tujuan utama dari pengujian ini adalah untuk memverifikasi bahwa sistem dapat mengenali status *tray* kosong atau berisi secara tepat, dan merespon sesuai dengan logika kendali sistem. Langkah-langkah pengujian dilakukan sebagai berikut:

1. Persiapan Sistem
  - Lima buah sensor infrared (IR) dipasang pada masing-masing posisi *tray* pada robot.
  - Sistem ESP32 yang membaca *input* dari sensor IR dikonfigurasi untuk mengirimkan *output* digital (HIGH/LOW) ke modul pengendali pintu.
2. Inisialisasi *Tray* Kosong
  - Seluruh *tray* dibiarkan dalam keadaan kosong.
  - Sistem dihidupkan untuk mendeteksi apakah seluruh sensor IR mengeluarkan output logika LOW.
  - Status logika ini kemudian digunakan untuk mengaktifkan aksi “pintu tertutup” secara otomatis oleh sistem.
3. Simulasi Kondisi Bertingkat
  - Pengujian dilanjutkan dengan menempatkan objek (gelas/kotak makanan) secara bertahap pada *tray*.
  - Setiap kombinasi dari 1 hingga 5 *tray* diisi secara berurutan.
  - Untuk tiap konfigurasi, sistem mencatat jumlah sensor IR yang aktif (logika HIGH) dan memastikan bahwa sistem merespon dengan membuka pintu apabila  $\Sigma \text{IR} \geq 1$ .

#### 4. Observasi dan Pencatatan Data

- Setiap konfigurasi pengisian *tray* dan output sensor diamati.
- Data output dari kelima sensor IR direkam dalam bentuk logika digital (1 = terdeteksi, 0 = tidak terdeteksi).
- Jumlah IR aktif ( $\Sigma$ IR), status *tray* (“Kosong” / “Berisi”), dan aksi pintu (“Tertutup” / “Terbuka”) dicatat dan dirangkum dalam tabel hasil pengujian.

#### 5. Validasi Logika Kontrol

Sistem dianggap berhasil apabila:

- Seluruh *tray* kosong menghasilkan  $\Sigma$ IR = 0 dan pintu tertutup.
- Minimal satu *tray* berisi menghasilkan  $\Sigma$ IR  $\geq$  1 dan pintu terbuka.
- Validasi ini menjadi dasar evaluasi integritas fungsi sensor terhadap logika kontrol aktual pada robot TIFA.

#### 6. Pengujian Berulang

- Untuk menjamin kestabilan sistem, setiap konfigurasi diuji sebanyak 3 kali dengan hasil yang konsisten.
- Tidak ditemukan anomali deteksi maupun *delay* pada pengiriman data ke ESP32.

Proses ini dirancang untuk menyerupai skenario operasional aktual di kafe, dan menjadi acuan dalam validasi sistem sebelum diimplementasikan penuh di Tel-U Coffee.

### I. Hasil Pengujian IR Sensor

Hasil pengujian terhadap kombinasi berbagai kondisi sensor ditampilkan pada tabel berikut.

**Tabel 5. 8 Hasil *Integration Testing* IR Sensor**

IR1	IR2	IR3	IR4	IR5	$\Sigma$ IR	Status <i>Tray</i>	Aksi Pintu
1	1	1	1	1	5	Berisi	Terbuka
0	1	1	1	1	4	Berisi	Terbuka
0	0	1	1	1	3	Berisi	Terbuka
0	0	0	1	1	2	Berisi	Terbuka
0	0	0	0	1	1	Berisi	Terbuka
0	0	0	0	0	0	Kosong	Tertutup

## J. Skenario *Voltage sensor*

Pengujian dilakukan dalam beberapa skenario utama untuk mensimulasikan kondisi nyata tegangan baterai pada sistem TIFA:

### 1. Skenario Tegangan Minimum

Tegangan *input* dari baterai berada di ambang batas bawah operasional sistem (sekitar 49.2V), yang telah *distep-down* menjadi  $\pm 22.7V$ . Sistem seharusnya mendeteksi kondisi ini sebagai baterai “hampir habis” dan menampilkan persentase sekitar 0%.

### 2. Skenario Tegangan Maksimum

Tegangan *input* dari baterai berada di puncak optimal (sekitar 53V), yang *distep-down* menjadi  $\pm 24.5V$ . Sistem seharusnya menampilkan persentase 100% sebagai kondisi baterai “penuh”.

### 3. Skenario Tegangan Menengah

Dilakukan pengujian pada beberapa titik tegangan antara minimum dan maksimum untuk memastikan pembacaan proporsional dan linier terhadap perubahan tegangan.

### 4. Skenario Pembacaan Berulang

Setiap konfigurasi diuji sebanyak 3 kali guna mengamati konsistensi hasil, kestabilan pembacaan sensor, serta memastikan tidak ada *Noise* atau *delay* dalam pembacaan data oleh ESP32.

## K. Proses Pengujian *Voltage sensor*

Langkah-langkah sistematis yang dilakukan dalam pengujian sensor tegangan adalah sebagai berikut:

### 1. Persiapan Sistem

- Sensor tegangan dipasang pada jalur output dari DC *step-down* converter yang menerima tegangan *input* dari baterai (hingga 53V) dan menurunkannya menjadi  $<25V$ .
- Sistem ESP32 dikonfigurasi untuk membaca tegangan melalui ADC internal atau sensor digital.
- Konversi dari nilai tegangan ke bentuk persentase dilakukan menggunakan rumus:

$$V_{ADC} = \frac{ADC_{Value} \times V_{ref}}{ADC_{max}}$$

$$\text{Divider Ratio} = \frac{V_{Aktual\ Baterai}}{V_{Masuk\ ESP}}$$

$$V_{Baterai} = V_{ADC} \times \text{Divider Ratio}$$

Jika  $voltMin = 49.2$  dan  $voltMax = 53.0$ , maka:

$$\text{StepPercent} = \frac{100}{\frac{53 - 49.2}{0.1}} = 2.65\%$$

$$\text{step} = \frac{V_{Baterai} - V_{Minimum}}{0.1}$$

$$\text{Percentage} = \text{step} \times 2.65\%$$

## 2. Inisialisasi Tegangan Kosong

- Sumber tegangan diatur pada 49.2V.
- Setelah melalui *step-down*, tegangan menjadi sekitar 22.7V dan dibaca oleh ESP32.
- Sistem diharapkan membaca nilai yang menunjukkan kondisi “baterai habis” dengan persentase mendekati 0%.

## 3. Simulasi Kondisi Bertingkat

- Tegangan *input* ditingkatkan secara bertahap: 50V, 51V, 52V, hingga 53V.
- Pembacaan sensor diamati pada tiap titik tegangan.
- Hasil pembacaan tegangan aktual dan konversi persentase dicatat dan dibandingkan dengan nilai teoritis.

## 4. Validasi Logika Konversi

- Pengujian dinyatakan valid apabila:
  - Tegangan 53.0V menghasilkan persentase  $\approx 100\%$
  - Tegangan 49.2V menghasilkan persentase  $\approx 0\%$
  - Nilai di antara keduanya berubah secara linier

- Tidak ditemukan lonjakan (*spike*), fluktuasi abnormal, atau *delay* dalam pembacaan data tegangan.

## L. Hasil Pengujian *Voltage sensor*

Hasil pengujian terhadap kombinasi berbagai kondisi sensor ditampilkan pada tabel berikut.

**Tabel 5.9 Hasil *Integration Testing Voltage sensor***

Percentase (%)	V Battery (V)	V ESP32 (V)	Raw ADC (12-bit)
100%	53	3.3125	4095
95%	52.8	3.3	4095
89%	52.6	3.2875	4071
84%	52.4	3.275	4047
79%	52.2	3.2625	4023
74%	52	3.25	3999
68%	51.8	3.2375	3975
63%	51.6	3.225	3951
58%	51.4	3.2125	3927
5%	49.4	3.0875	3687
0%	49.2	3.075	3663

### 5.2.3 *Alpha Testing*

Pada tahap ini, pengujian dilakukan oleh tim pengembang dengan pendekatan *black-box Testing* untuk memastikan integrasi antar modul sistem berjalan dengan baik. Pengujian ini menitikberatkan pada alur interaksi antar komponen utama aplikasi dan web monitoring. Dengan kata lain, pengujian berfokus pada keluaran sistem yang dihasilkan terhadap masukan tertentu. Tahapan ini bertujuan untuk memastikan bahwa seluruh fungsi yang saling bergantung telah terintegrasi dengan benar sebelum sistem dilanjutkan ke tahap pengujian eksternal.

### A. Skenario Detil Aplikasi

Tabel ini mencakup skenario pengujian menggunakan jenis *Black Box* untuk memverifikasi fungsi utama aplikasi TIFA pada berbagai halaman yang merupakan fitur inti berdasarkan dokumen.

**Tabel 5. 9 Skenario *Alpha Testing* Aplikasi Mode Antar**

No.	Halaman	Aksi yang Diuji	Detail Pengujian
1.	Home	Memilih tombol ‘Antar’	Memilih tombol ‘Antar’ untuk perpindahan halaman
2.	Antar	Membuka pintu robot	Menekan tombol buka pintu untuk membuka pintu
3.	Antar	Menutup pintu robot	Menekan tombol tutup pintu untuk menutup pintu
4.	Antar	Memilih <i>tray</i>	Menekan tombol <i>tray</i> posisi yang dipilih
5.	Antar	Memilih nomor meja	Menekan tombol nomor meja yang dipilih
6.	Antar	Menambahkan pesanan	Menekan tombol tambah pesanan dari hasil pilihan <i>tray</i> dan nomor meja di tambah
7.	Antar	Melihat semua pesanan	Menekan tombol oke untuk melihat semua pesanan yang dipilih
8.	Antar	Menghapus pesanan	Menekan tombol ikon sampah pada pesanan yang salah
9.	Antar	Memulai pengiriman	Menekan tombol konfirmasi pesanan untuk memulai pengantaran
10.	Loading	Konfirmasi pesanan	Menekan tombol konfirmasi pesanan sesuai dengan informasi pesanan

**Tabel 5. 10 Skenario Pengolahan Nomor Meja**

No.	Halaman	Aksi yang Diuji	Detail Pengujian
1.	Antar	Menambahkan nomor meja	Menekan tombol tambah meja untuk menambahkan nomor meja

No.	Halaman	Aksi yang Diuji	Detail Pengujian
2.	Antar	Memasukkan nomor meja	Memasukkan nama nomor meja dan nomor meja sesuai yang di inginkan dan tersedia contohnya
3.	Antar	Melihat nomor meja yang ingin dihapus	Menekan tombol hapus meja untuk melihat semua pilihan nomor meja yang bisa di hapus
4.	Antar	Menghapus nomor meja	Menekan nomor meja yang ingin di hapus lalu tekan tombol hapus

**Tabel 5. 11 Skenario Pengolahan Media**

No.	Halaman	Aksi yang Diuji	Detail Pengujian
1.	Home	Memilih tombol ‘Media’	Memilih tombol ‘Media’ untuk perpindahan halaman
2.	Media	Menambahkan foto	Menekan tombol tambah foto untuk masuk ke penyimpanan foto yang dipilih
3.	Media	Menambahkan video	Menekan tombol tambah video untuk masuk ke penyimpanan video yang dipilih
4.	Media	Menghapus media	Menekan tombol hapus item untuk menghapus media terakhir

**Tabel 5. 12 Skenario Fitur Peta**

No.	Halaman	Aksi yang Diuji	Detail Pengujian
1.	Home	Memilih tombol ‘Peta’	Memilih tombol ‘Peta’ untuk perpindahan halaman dan melihat semua peta yang tersimpan

**Tabel 5. 13 Skenario Fitur *TalkBack***

No.	Halaman	Aksi yang Diajukan	Detail Pengujian
1.	Home	Memilih tombol ‘T.I.F.A’	Memilih tombol ‘T.I.F.A’ untuk membuat aktif fitur <i>TalkBack</i> nya

## B. Proses Pengujian Aplikasi

Proses pengujian alpha untuk aplikasi TIFA dilakukan secara sistematis untuk memastikan semua fitur diajukan dengan baik. Berikut langkah-langkahnya:

### 1. Persiapan:

- *Install* aplikasi TIFA pada perangkat yang kompatibel.
- Pastikan perangkat terhubung ke jaringan Tel-U Coffee yang sama dan robot siap dioperasikan.

### 2. Pelaksanaan Pengujian:

Untuk setiap skenario pengujian yang telah ditentukan.

- Buka bagian aplikasi yang sesuai dengan skenario yang sedang diajukan.
- Lakukan tindakan yang diajukan sebagaimana dijelaskan dalam skenario.
- Amati perilaku aplikasi dan bandingkan dengan hasil yang diharapkan.
- Catat hasil aktual dari pengujian dan tentukan status keberhasilan (berhasil atau gagal) berdasarkan perbandingan tersebut.

### 3. Pelaporan Bug:

Jika ada perbedaan antara hasil aktual dan yang diharapkan, dokumentasikan masalahnya.

### 4. Pengujian Ulang:

Setelah perbaikan dilakukan, ulangi pengujian pada skenario yang gagal untuk memastikan masalah telah terselesaikan.

## C. Hasil Pengujian Aplikasi

Berdasarkan pengujian alpha yang dilakukan, seluruh fitur utama aplikasi menunjukkan hasil yang sesuai dengan ekspektasi. Setiap aksi yang diajukan memberikan respon sistem yang tepat dan berjalan tanpa kendala. Hasil pengamatan menunjukkan bahwa sistem mampu

merespon *input* pengguna secara akurat, sehingga dapat disimpulkan bahwa fungsi-fungsi utama telah berjalan dengan baik. Rincian hasil pengujian disajikan dalam tabel berikut.

**Tabel 5. 14 Hasil Pengujian *Alpha Testing Mode* Antar**

No.	Aksi yang Diuji	Hasil yang Diharapkan	Hasil Pengamatan	Kesimpulan
1.	Memilih tombol ‘Antar’	Memilih tombol ‘Antar’ untuk perpindahan halaman	Halaman berpindahan	Berhasil
2.	Membuka pintu robot	Menekan tombol buka pintu untuk membuka pintu	Pintu terbuka	Berhasil
3.	Menutup pintu robot	Menekan tombol tutup pintu untuk menutup pintu	Pintu tertutup	Berhasil
4.	Memilih <i>tray</i>	Menekan tombol <i>tray</i> posisi yang dipilih	<i>Tray</i> posisi dapat dipilih	Berhasil
5.	Memilih nomor meja	Menekan tombol nomor meja yang dipilih	Nomor meja dapat dipilih	Berhasil
6.	Menambahkan pesanan	Menekan tombol tambah pesanan dari hasil pilihan <i>tray</i> dan nomor meja ditambah	<i>Tray</i> dan nomor meja dapat ditambah	Berhasil
7.	Melihat semua pesanan	Menekan tombol oke untuk melihat semua pesanan yang dipilih	Dapat melihat semua pesanan yang dipilih	Berhasil
8.	Menghapus pesanan	Menekan tombol ikon sampah pada pesanan yang salah	Dapat menghapus pada pesanan yang salah	Berhasil
9.	Memulai pengiriman	Menekan tombol konfirmasi pesanan untuk memulai pengantaran	Dapat menekan tombol konfirmasi pesanan	Berhasil

No.	Aksi yang Diuji	Hasil yang Diharapkan	Hasil Pengamatan	Kesimpulan
10.	Konfirmasi pesanan	Menekan tombol konfirmasi pesanan sesuai dengan informasi pesanan	Dapat mengkonfirmasi pesanan yang sesuai	Berhasil

**Tabel 5. 15 Hasil Pengujian *Alpha Testing* Pengolahan Nomor Meja**

No.	Aksi yang Diuji	Hasil yang Diharapkan	Hasil Pengamatan	Kesimpulan
1.	Menambahkan nomor meja	Menekan tombol tambah meja untuk menambahkan nomor meja	Nomor meja dapat ditambah	Berhasil
2.	Memasukkan nomor meja	Memasukkan nama nomor meja dan nomor meja sesuai yang diinginkan dan tersedia contohnya	Nama nomor meja dan nomor meja sesuai dapat ditambah	Berhasil
3.	Melihat nomor meja yang ingin dihapus	Menekan tombol hapus meja untuk melihat semua pilihan nomor meja yang bisa dihapus	Nomor meja yang ingin dihapus dapat tampilan	Berhasil
4.	Menghapus nomor meja	Menekan nomor meja yang ingin dihapus lalu tekan tombol hapus	Nomor meja dapat dihapus	Berhasil

**Tabel 5. 16 Hasil Pengujian *Alpha Testing* Pengolahan Media**

No.	Aksi yang Diuji	Hasil yang Diharapkan	Hasil Pengamatan	Kesimpulan
1.	Memilih tombol ‘Media’	Memilih tombol ‘Media’ untuk perpindahan halaman	Halaman berpindah ke bagian media	Berhasil

No.	Aksi yang Diuji	Hasil yang Diharapkan	Hasil Pengamatan	Kesimpulan
2.	Menambahkan foto	Menekan tombol tambah foto untuk masuk ke penyimpanan foto yang dipilih	Dapat memilih dan menambahkan foto	Berhasil
3.	Menambahkan video	Menekan tombol tambah video untuk masuk ke penyimpanan video yang dipilih	Dapat memilih dan menambahkan video	Berhasil
4.	Menghapus media	Menekan tombol hapus item untuk menghapus media terakhir	Media terakhir berhasil dihapus	Berhasil

**Tabel 5. 17 Hasil Pengujian *Alpha Testing* Fitur Peta**

No.	Aksi yang Diuji	Hasil yang Diharapkan	Hasil Pengamatan	Kesimpulan
1.	Memilih tombol ‘Peta’	Memilih tombol ‘Peta’ untuk perpindahan halaman dan melihat semua peta yang tersimpan	Halaman berhasil berpindah dan semua peta yang tersimpan ditampilkan	Berhasil

**Tabel 5. 18 Hasil Pengujian *Alpha Testing* Fitur *TalkBack***

No.	Aksi yang Diuji	Hasil yang Diharapkan	Hasil Pengamatan	Kesimpulan
1.	Memilih tombol ‘T.I.F.A’	Memilih tombol ‘T.I.F.A’ untuk membuat aktif fitur <i>TalkBack</i> nya	Fitur <i>TalkBack</i> berhasil diaktifkan saat tombol ditekan	Berhasil

#### D. Skenario Detil Web Monitoring

Tabel ini mencakup skenario pengujian menggunakan jenis *Black Box* untuk memverifikasi fungsi utama web monitoring pada berbagai komponen/element dan halaman yang merupakan fitur inti berdasarkan dokumen.

**Tabel 5. 19 Skenario *Alpha Testing* Web Monitoring Halaman *Login***

No.	Halaman	Aksi yang Diuji	Detail Pengujian
1.	<i>Login</i>	Klik <i>input field</i> “Username”	Klik <i>input field</i> “Username” untuk memasukkan <i>username</i> yang terdaftar
2.	<i>Login</i>	Klik <i>input field</i> “Password”	Klik <i>input field</i> “Password” untuk memasukkan <i>password</i> yang terdaftar
3.	<i>Login</i>	Klik tombol “Login”	Klik tombol “Login” agar halaman ternavigasi ke halaman <i>Dashboard</i>

**Tabel 5. 20 Skenario *Alpha Testing* Web Monitoring Elemen Header**

No.	Elemen	Aksi yang Diuji	Detail Pengujian
1.	<i>Header</i>	Klik ikon lonceng	Klik ikon lonceng untuk melihat <i>Pop up</i> notifikasi
2.	<i>Pop-up</i> notifikasi	Klik satu notifikasi	Klik satu notifikasi yang ada di <i>Pop up</i> untuk diarahkan ke halaman yang sesuai dengan notifikasinya
3.	<i>Pop-up</i> notifikasi	Klik tombol “Tandai Semua Terbaca”	Klik tombol “Tandai Semua Terbaca” untuk memberikan tanda bahwa semua notif telah dibaca
4.	<i>Header</i>	Klik avatar/ <i>initial user</i>	Klik avatar/ <i>initial user</i> untuk membuka <i>dropdown</i>
5.	<i>Dropdown Logout</i>	Klik tombol <i>Logout</i>	Klik “Logout” dan pastikan pengguna diarahkan kembali ke halaman <i>login</i>

**Tabel 5. 21 Skenario *Alpha Testing* Web Monitoring Elemen Sidebar**

No.	Elemen	Aksi yang Diuji	Detail Pengujian
1.	<i>Sidebar</i>	Klik menu navigasi “Dashboard”	Klik menu navigasi “Dashboard” untuk menuju ke halaman <i>Dashboard</i>
2.	<i>Sidebar</i>	Klik menu navigasi “Daftar Robot”	Klik menu navigasi “Daftar Robot” untuk menuju ke halaman daftar robot
3.	<i>Sidebar</i>	Klik menu navigasi “Tambah Robot”	Klik menu navigasi “Tambah Robot” untuk menuju ke halaman tambah robot
4.	<i>Sidebar</i>	Klik menu navigasi “Tambah Jenis Robot”	Klik menu navigasi “Tambah Jenis Robot” untuk menuju ke halaman tambah jenis robot
5.	<i>Sidebar</i>	Klik tombol “Logout”	Klik tombol “Logout” untuk mengakhiri sesi dan kembali ke halaman saat pertama web diakses (halaman <i>login</i> )

**Tabel 5. 22 Skenario *Alpha Testing* Web Monitoring Halaman Dashboard**

No.	Halaman	Aksi yang Diuji	Detail Pengujian
1.	<i>Dashboard</i>	Klik tab “Hari” pada grafik aktivitas robot	Klik tab “Hari” pada grafik aktivitas robot untuk menampilkan grafik aktivitas robot setiap hari

No.	Halaman	Aksi yang Diuji	Detail Pengujian
2.	<i>Dashboard</i>	Klik tab “Minggu” pada grafik aktivitas robot	Klik tab “Minggu” pada grafik aktivitas robot untuk menampilkan grafik aktivitas robot setiap minggu
3.	<i>Dashboard</i>	Klik tab “Bulan” pada grafik aktivitas robot	Klik tab “Bulan” pada grafik aktivitas robot untuk menampilkan grafik aktivitas robot setiap bulan
4.	<i>Dashboard</i>	Klik tab “Hari” pada grafik banyaknya pesanan per hari	Klik tab “Hari” pada grafik banyaknya pesanan per hari untuk menampilkan grafik banyaknya pesanan per hari setiap hari
5.	<i>Dashboard</i>	Klik tab “Minggu” pada grafik banyaknya pesanan per hari	Klik tab “Minggu” pada grafik banyaknya pesanan per hari untuk menampilkan grafik banyaknya pesanan per hari setiap minggu
6.	<i>Dashboard</i>	Klik tab “Bulan” pada grafik banyaknya pesanan per hari	Klik tab “Bulan” pada grafik banyaknya pesanan per hari untuk menampilkan grafik banyaknya pesanan per hari setiap bulan

**Tabel 5. 23 Skenario Alpha Testing Web Monitoring Halaman Daftar Robot**

No.	Halaman/Elemen	Aksi yang Diuji	Detail Pengujian
1.	Daftar Robot	Klik tombol “Detail” pada salah satu robot yang terdaftar	Klik tombol “Detail” pada salah satu robot yang terdaftar untuk menuju ke halaman Detail robot
2.	Daftar Robot	Klik tombol “Metrik” pada salah satu robot yang terdaftar	Klik tombol “Metrik” pada salah satu robot yang terdaftar untuk menuju ke halaman metrik
3.	Daftar Robot	Klik tombol “Edit” pada salah satu robot yang terdaftar	Klik tombol “Edit” pada salah satu robot yang terdaftar dan akan muncul <i>Pop up</i> edit robot
4.	Detail Robot	Klik tombol “Lihat Semua” pada log aktivitas robot <i>card</i>	Klik tombol “Lihat Semua” pada log aktivitas robot <i>card</i> untuk menampilkan <i>Pop-up</i> berisi log aktivitas robot secara lengkap
5.	Detail Robot	Klik tombol “Lihat Semua” pada log aktivitas sensor <i>card</i>	Klik tombol “Lihat Semua” pada log aktivitas sensor <i>card</i> untuk menampilkan <i>Pop-up</i> berisi log aktivitas sensor secara lengkap
6.	Detail Robot	Klik tombol “Tutup” pada <i>Pop-up</i> log aktivitas robot atau log aktivitas sensor	Klik tombol “Tutup” pada <i>Pop-up</i> log aktivitas robot atau log aktivitas sensor untuk kembali ke halaman sebelumnya (halaman Detail robot)

No.	Halaman/Elemen	Aksi yang Diuji	Detail Pengujian
7.	Metrik	Klik tab “Hari, Minggu, Bulan” pada setiap grafik	Klik tab “Hari, Minggu, Bulan” pada setiap grafik untuk menampilkan data grafik selama sehari, seminggu, dan sebulan
8.	<i>Pop-up</i> Edit Robot	Klik <i>input field</i> “Serial Robot”	Klik <i>input field</i> “Serial Robot” untuk mengedit serial robot
9.	<i>Pop-up</i> Edit Robot	Klik <i>dropdown</i> “Jenis Robot”	Klik <i>dropdown</i> “Jenis Robot” untuk menampilkan daftar jenis robot yang sudah terdaftar dan dapat dipilih
10.	<i>Pop-up</i> Edit Robot	Klik tombol “Simpan Perubahan”	Klik tombol “Simpan Perubahan” untuk menyimpan data baru
11.	<i>Pop-up</i> Edit Robot	Klik tombol “Batal”	Klik tombol “Batal” untuk menutup <i>Pop-up</i> dan tidak ada data yang berubah
12.	<i>Pop-up</i> Edit Robot	Klik ikon “X” di pojok kanan atas	Klik ikon “X” di pojok kanan atas untuk menutup <i>Pop-up</i> dan tidak ada data yang berubah

**Tabel 5. 24 Skenario *Alpha Testing* Web Monitoring Halaman Tambah Robot**

No.	Halaman	Aksi yang Diuji	Detail Pengujian
1.	Tambah Robot	Klik <i>input field</i> “Nama Robot”	Klik <i>input field</i> “Nama Robot” untuk memasukkan nama robot

No.	Halaman	Aksi yang Diuji	Detail Pengujian
2.	Tambah Robot	Klik <i>input field</i> “Nomor Seri”	Klik <i>input field</i> “Nomor Seri” untuk memasukkan nomor seri robot
3.	Tambah Robot	Klik <i>dropdown</i> “Jenis Robot”	Klik <i>dropdown</i> “Jenis Robot” untuk memilih jenis robot yang sudah terdaftar
4.	Tambah Robot	Klik tombol “Simpan”	Klik tombol “Simpan” agar data robot baru tersimpan

**Tabel 5. 25 Skenario *Alpha Testing* Web Monitoring Halaman Tambah Jenis Robot**

No.	Halaman	Aksi yang Diuji	Detail Pengujian
1.	Tambah Jenis Robot	Klik <i>input field</i> “Nama Jenis Robot”	Klik <i>input field</i> “Nama Jenis Robot” untuk memasukkan nama jenis robot baru yang akan ditambahkan
2.	Tambah Jenis Robot	Klik tombol “Simpan”	Klik tombol “Simpan” agar data jenis robot baru tersimpan

## E. Proses Pengujian Web Monitoring

Proses pengujian alpha terhadap sistem web monitoring TIFA dilaksanakan secara sistematis guna memastikan bahwa seluruh fitur antarmuka berjalan sesuai dengan spesifikasi yang telah dirancang. Pengujian ini juga bertujuan untuk mengevaluasi stabilitas sistem, akurasi tampilan data, serta *responsivitas* antarmuka pengguna.

### 1. Persiapan

- Pastikan sistem *backend* (API dan database MySQL) telah berjalan dan terhubung dengan web monitoring.
- Buka web monitoring TIFA pada *browser* yang kompatibel.

- Pastikan perangkat pengakses web berada dalam jaringan yang sama dengan sistem TIFA, untuk menjamin koneksi *real-time* dengan data robot.

## 2. Pelaksanaan Pengujian

Untuk setiap skenario pengujian yang telah ditentukan:

- Akses halaman web sesuai dengan fitur yang diuji.
- Lakukan interaksi pengguna yang relevan, seperti:
  - Menavigasi halaman *Dashboard*.
  - Mengamati perubahan status robot secara *real-time*.
  - Memeriksa apakah data pada web sesuai dengan data di database.
- Memeriksa *respons* dan tampilan dari web monitoring, lalu bandingkan dengan hasil yang diharapkan dari skenario pengujian.
- Mencatat hasil aktual dari setiap pengujian dan berikan status keberhasilan (berhasil atau gagal) sesuai dengan hasil perbandingan.

## 3. Pengujian Ulang

- Setelah dilakukan perbaikan pada bagian yang gagal, lakukan pengujian ulang dengan mengikuti skenario yang sama.
- Verifikasi bahwa masalah telah terselesaikan dan tidak menimbulkan bug baru di bagian lain dari sistem.

## F. Hasil Pengujian Web Monitoring

Berdasarkan pengujian alpha yang dilakukan, seluruh fitur utama web monitoring menunjukkan hasil yang sesuai dengan ekspektasi. Setiap aksi yang diuji memberikan respon sistem yang tepat dan berjalan tanpa kendala. Hasil pengamatan menunjukkan bahwa sistem mampu merespon *input* pengguna secara akurat, sehingga dapat disimpulkan bahwa fungsi-fungsi utama telah berjalan dengan baik. Rincian hasil pengujian disajikan dalam tabel berikut.

**Tabel 5. 26 Hasil Pengujian Alpha Testing Halaman Login**

No.	Aksi yang Diuji	Hasil yang Diharapkan	Hasil Pengamatan	Kesimpulan
1.	Klik <i>input field</i> “Username”	Kursor aktif dan <i>input field</i> “Username” dapat diisi	Kursor aktif dan <i>input field</i> “Username” dapat diisi	Berhasil

No.	Aksi yang Diuji	Hasil yang Diharapkan	Hasil Pengamatan	Kesimpulan
2.	Klik <i>input field</i> “Password”	Kursor aktif dan <i>input field</i> “Password” dapat diisi	Kursor aktif dan <i>input field</i> “Password” dapat diisi	Berhasil
3.	Klik tombol “Login”	Menampilkan halaman <i>Dashboard</i>	Dapat menampilkan halaman <i>Dashboard</i>	Berhasil

**Tabel 5. 27 Hasil Pengujian Alpha Testing Elemen Header**

No.	Aksi yang Diuji	Hasil yang Diharapkan	Hasil Pengamatan	Kesimpulan
1.	Klik ikon lonceng	<i>Pop up</i> notifikasi muncul dan menampilkan daftar notifikasi	<i>Pop up</i> notifikasi muncul dengan daftar notifikasi yang tersedia	Berhasil
2.	Klik satu notifikasi	Sistem navigasi pengguna ke halaman sesuai dengan isi notifikasi	Halaman yang dituju sesuai dengan notifikasi yang diklik	Berhasil
3.	Klik tombol “Tandai Semua Terbaca”	Semua notifikasi diubah statusnya menjadi terbaca dan tampilan diperbarui	Status notifikasi berubah menjadi terbaca setelah tombol diklik	Berhasil
4.	Klik avatar/ <i>initial user</i>	<i>Dropdown</i> muncul setelah avatar diklik	<i>Dropdown</i> berhasil tampil setelah avatar diklik	Berhasil
5.	Klik tombol <i>Logout</i>	Pengguna diarahkan kembali ke halaman <i>login</i> dan sesi berakhir	Sistem berhasil keluar dan membawa pengguna ke halaman <i>login</i>	Berhasil

**Tabel 5. 28 Hasil Pengujian *Alpha Testing* Elemen Sidebar**

No.	Aksi yang Diuji	Hasil yang Diharapkan	Hasil Pengamatan	Kesimpulan
1.	Klik menu navigasi “ <i>Dashboard</i> ”	Halaman berpindah ke tampilan <i>Dashboard</i> utama	Sistem menampilkan halaman <i>Dashboard</i> setelah menu diklik	Berhasil
2.	Klik menu navigasi “Daftar Robot”	Halaman berpindah ke halaman daftar robot	Navigasi berhasil membawa pengguna ke halaman daftar robot	Berhasil
3.	Klik menu navigasi “Tambah Robot”	Halaman berpindah ke halaman tambah robot	Navigasi berhasil membawa pengguna ke halaman tambah robot	Berhasil
4.	Klik menu navigasi “Tambah Jenis Robot”	Halaman berpindah ke halaman tambah jenis robot	Navigasi berhasil membawa pengguna ke halaman tambah jenis robot	Berhasil
5.	Klik tombol “ <i>Logout</i> ”	Pengguna diarahkan kembali ke halaman <i>login</i> dan sesi berakhir	Pengguna otomatis keluar dan dibawa ke halaman <i>login</i>	Berhasil

**Tabel 5. 29 Hasil Pengujian *Alpha Testing* Halaman Dashboard**

No.	Aksi yang Diuji	Hasil yang Diharapkan	Hasil Pengamatan	Kesimpulan
1.	Klik tab “Hari” pada grafik aktivitas robot	Grafik menampilkan aktivitas harian	Tampilan grafik berhasil menunjukkan aktivitas robot setiap hari	Berhasil

No.	Aksi yang Diuji	Hasil yang Diharapkan	Hasil Pengamatan	Kesimpulan
		dari robot yang dipantau		
2.	Klik tab “Minggu” pada grafik aktivitas robot	Grafik memperlihatkan aktivitas robot perminggu	Data grafik menyesuaikan dan menampilkan aktivitas robot perminggu	Berhasil
3.	Klik tab “Bulan” pada grafik aktivitas robot	Sistem menyajikan grafik aktivitas robot perbulan	Grafik memuat data aktivitas robot perbulan	Berhasil
4.	Klik tab “Hari” pada grafik banyaknya pesanan per hari	Grafik menampilkan total pesanan setiap hari	Grafik menampilkan jumlah pesanan setiap hari dengan benar	Berhasil
5.	Klik tab “Minggu” pada grafik banyaknya pesanan per hari	Sistem menampilkan grafik jumlah pesanan perminggu	Grafik berhasil menampilkan jumlah pesanan perminggu	Berhasil
6.	Klik tab “Bulan” pada grafik banyaknya pesanan per hari	Sistem memuat grafik jumlah pesanan setiap bulan	Grafik menampilkan total pesanan setiap bulan dengan benar	Berhasil

**Tabel 5. 30 Hasil Pengujian *Alpha Testing* Halaman Daftar Robot**

No.	Aksi yang Diuji	Hasil yang Diharapkan	Hasil Pengamatan	Kesimpulan
1.	Klik tombol “Detail” pada salah satu robot yang terdaftar	Sistem menampilkan halaman Detail robot yang dipilih	Halaman Detail robot terbuka sesuai Detail robot yang dipilih	Berhasil
2.	Klik tombol “Metrik” pada salah satu robot yang terdaftar	Halaman metrik robot terbuka menampilkan grafik	Sistem berhasil menavigasi ke halaman metrik dan menampilkan grafik	Berhasil
3.	Klik tombol “Edit” pada salah satu robot yang terdaftar	Muncul <i>Pop-up</i> edit robot dengan <i>field</i> yang terisi sesuai data awal	<i>Pop-up</i> muncul dan menampilkan data robot sesuai dengan robot yang dipilih	Berhasil
4.	Klik tombol “Lihat Semua” pada log aktivitas robot <i>card</i>	<i>Pop-up</i> log aktivitas robot muncul dan menampilkan log secara lengkap	Log aktivitas robot lengkap berhasil ditampilkan dalam tampilan <i>Pop-up</i>	Berhasil
5.	Klik tombol “Lihat Semua” pada log aktivitas sensor <i>card</i>	<i>Pop-up</i> log aktivitas sensor muncul dan menampilkan log secara lengkap	Log aktivitas sensor lengkap berhasil ditampilkan dalam tampilan <i>Pop-up</i>	Berhasil

No.	Aksi yang Diuji	Hasil yang Diharapkan	Hasil Pengamatan	Kesimpulan
6.	Klik tombol “Tutup” pada <i>Pop-up</i> log aktivitas robot atau log aktivitas sensor	<i>Pop-up</i> tertutup dan kembali ke halaman Detail robot	<i>Pop-up</i> dapat tertutup dan kembali ke halaman Detail robot	Berhasil
7.	Klik tab “Hari, Minggu, Bulan” pada setiap grafik	Grafik menyesuaikan tampilan berdasarkan tab waktu yang dipilih	Grafik dapat menyesuaikan tampilan berdasarkan tab waktu yang dipilih	Berhasil
8.	Klik <i>input field</i> “Serial Robot”	<i>Field</i> aktif dan bisa diubah sesuai kebutuhan pengguna	<i>Field</i> aktif dan dapat diubah sesuai kebutuhan	Berhasil
9.	Klik <i>dropdown</i> “Jenis Robot”	<i>Dropdown</i> terbuka dan menampilkan daftar jenis robot yang terdaftar serta dapat dipilih	<i>Dropdown</i> terbuka dan menampilkan daftar jenis robot yang terdaftar serta dapat dipilih	Berhasil
10.	Klik tombol “Simpan Perubahan”	Perubahan data tersimpan dan <i>Pop-up</i> tertutup	Perubahan data dapat tersimpan dan <i>Pop-up</i> tertutup	Berhasil
11.	Klik tombol “Batal”	<i>Pop-up</i> tertutup tanpa menyimpan perubahan	<i>Pop-up</i> dapat tertutup tanpa menyimpan perubahan	Berhasil
12.	Klik ikon “X” di pojok kanan atas	<i>Pop-up</i> ditutup tanpa menyimpan perubahan apa pun	<i>Pop-up</i> dapat ditutup tanpa menyimpan perubahan apa pun	Berhasil

**Tabel 5. 31 Hasil Pengujian *Alpha Testing* Halaman Tambah Robot**

No.	Aksi yang Diuji	Hasil yang Diharapkan	Hasil Pengamatan	Kesimpulan
1.	Klik <i>input field</i> “Nama Robot”	<i>Input field</i> “Nama Robot” dapat diklik dan diisi	<i>Input field</i> “Nama Robot” bisa diklik dan diisi sesuai kebutuhan	Berhasil
2.	Klik <i>input field</i> “Nomor Seri”	<i>Input field</i> “Nomor Seri” dapat diisi	<i>Input field</i> “Nomor Seri” dapat diisi	Berhasil
3.	Klik <i>dropdown</i> “Jenis Robot”	<i>Dropdown</i> terbuka dan menampilkan pilihan jenis robot yang sudah terdaftar dan dapat dipilih	<i>Dropdown</i> terbuka dan menampilkan pilihan jenis robot yang sudah terdaftar dan dapat dipilih	Berhasil
4.	Klik tombol “Simpan”	Data robot tersimpan dan halaman kembali/ter-reset setelah <i>submit</i>	Data robot tersimpan dan halaman kembali/ter-reset setelah <i>submit</i>	Berhasil

**Tabel 5. 32 Hasil Pengujian *Alpha Testing* Halaman Tambah Jenis Robot**

No.	Aksi yang Diuji	Hasil yang Diharapkan	Hasil Pengamatan	Kesimpulan
1.	Klik <i>input field</i> “Nama Jenis Robot”	<i>Input field</i> aktif dan bisa diisi dengan teks sesuai nama jenis robot	Pengguna dapat mengetikkan nama jenis robot pada <i>input field</i> “Nama Jenis Robot”	Berhasil

No.	Aksi yang Diuji	Hasil yang Diharapkan	Hasil Pengamatan	Kesimpulan
2.	Klik tombol “Simpan”	Sistem menyimpan data dan halaman form kembali kosong setelah <i>submit</i>	Data jenis robot baru tersimpan dan form kembali dalam kondisi awal	Berhasil

#### 5.2.4 User Acceptance Testing (UAT)

User Acceptance Testing (UAT) terhadap sistem TIFA dilakukan dengan metode survei untuk mendapatkan umpan balik langsung dari pengguna yang terlibat dalam operasional sehari-hari. Survei ini diisi oleh 6 orang barista yang berkerja di Tel-U Coffe untuk aplikasi, yang seluruhnya merupakan barista aktif di Tel-U Coffee. Mereka diminta untuk menilai pengalaman penggunaan aplikasi dan web TIFA berdasarkan beberapa aspek yang telah ditentukan. Penilaian dilakukan menggunakan skala Likert lima tingkat, yaitu Sangat Setuju (SS), Setuju (S), Ragu-ragu (RR), Tidak Setuju (TS), dan Sangat Tidak Setuju (STS). Masing-masing kategori diberikan bobot nilai tersendiri untuk keperluan analisis kuantitatif. Selain itu, satu orang administrator turut menjadi responden untuk memberikan penilaian terhadap sistem web monitoring, menggunakan skema penilaian yang sama.

**Tabel 5. 33 Bobot Nilai Setiap Kategori**

Jawaban	Bobot
SS : Sangat Setuju	5
S : Setuju	4
RR : Ragu - Ragu	3
TS : Tidak Setuju	2
STS : Sangat Tidak Setuju	1

#### A. Skenario Detil Aplikasi

Bagian ini memuat daftar pertanyaan yang disusun untuk mengevaluasi pengalaman pengguna saat berinteraksi dengan aplikasi TIFA. Kuesioner dirancang menggunakan Skala Likert dengan nilai 1 sampai 5, guna mengukur tingkat kemudahan penggunaan, kejelasan antarmuka, dan efektivitas sistem dalam mendukung proses kerja barista. Hasil dari pengisian

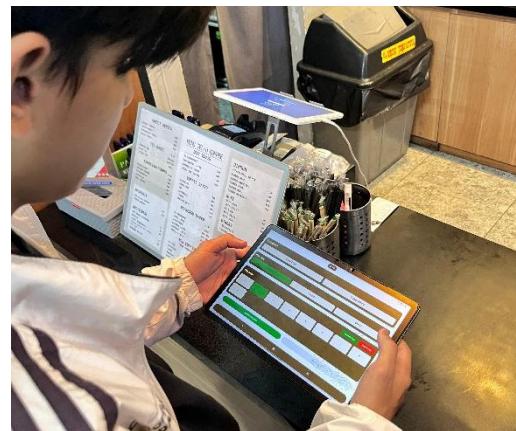
tabel ini akan digunakan sebagai dasar analisis dalam pengujian *User Acceptance Testing* (UAT).

**Tabel 5. 34 Skenario UAT Aplikasi**

No.	Pertanyaan	Nilai				
		1	2	3	4	5
1.	Apakah Anda dapat memahami fungsi dari setiap tombol pada halaman utama aplikasi TIFA tanpa bantuan petunjuk?					
2.	Apakah proses pemilihan <i>tray</i> dan nomor meja dapat dilakukan dengan mudah dan tanpa kebingungan?					
3.	Apakah aplikasi memberikan umpan balik ( <i>feedback</i> ) yang jelas setelah Anda menekan tombol “Start” pengantar?					
4.	Apakah Anda merasa aplikasi cukup <i>responsif</i> (tidak lambat atau <i>freeze</i> ) selama digunakan?					
5.	Apakah tampilan antarmuka (UI) aplikasi TIFA mudah dibaca dan tidak membingungkan?					
6.	Apakah Anda pernah mengalami <i>error</i> atau aplikasi tertutup sendiri ( <i>crash</i> ) saat melakukan <i>input</i> data?					
7.	Apakah Anda merasa <i>input tray</i> dan nomor meja aman dari kesalahan (misalnya tidak bisa klik jika kosong)?					
8.	Apakah Anda merasa fitur-fitur aplikasi ini membantu mempercepat proses pengiriman pesanan di jam sibuk?					
9.	Sejauh ini, apakah Anda merasa puas dengan pengalaman penggunaan aplikasi TIFA?					

## B. Proses Pengujian Aplikasi

Evaluasi sistem dilaksanakan secara offline di Tel-U Coffee, sebuah lingkungan operasional nyata yang memberikan konteks asli untuk pengujian aplikasi. Pengujian melibatkan Khaerul Muhammad Khadafi, seorang barista berpengalaman yang bertindak sebagai responden utama dalam evaluasi ini. Keterlibatan pengguna profesional dari industri kopi memberikan perspektif yang berharga mengenai kegunaan dan relevansi aplikasi robot TIFA dalam konteks pekerjaan sehari-hari.



**Gambar 5. 1 Proses Pengujian pada Barista**

Tujuan utama dari pengujian ini adalah memvalidasi fungsionalitas dan kinerja aplikasi yang terintegrasi pada robot TIFA dalam kondisi operasional sebenarnya. Proses evaluasi menggunakan pendekatan campuran yang terdiri dari demonstrasi langsung dan sesi uji coba praktis untuk mengamati interaksi pengguna dengan sistem secara *real-time*, serta pengumpulan data melalui kuesioner terstruktur yang dirancang untuk mengukur tingkat kepuasan dan pengalaman pengguna. Hasil evaluasi ini akan menjadi dasar untuk validasi kesiapan teknologi dan penyempurnaan sistem di masa mendatang.

### C. Hasil Pengujian Aplikasi

*User Acceptance Testing* (UAT) dilakukan untuk memperoleh penilaian dari pengguna terkait kemudahan, kejelasan, dan *respons* sistem dalam aplikasi TIFA. Berdasarkan tanggapan pengguna terhadap sejumlah pertanyaan, diperoleh hasil yang sangat positif. Hal ini menunjukkan bahwa aplikasi telah mampu memberikan pengalaman pengguna yang baik dan sesuai harapan. Rincian nilai dapat dilihat pada tabel berikut.

**Tabel 5. 35 Hasil UAT Aplikasi**

No.	Pertanyaan	STS	TS	RR	S	SS	Total Perhitungan	Nilai Rata - Rata
1.	Apakah Anda dapat memahami fungsi dari setiap tombol pada halaman utama aplikasi TIFA tanpa bantuan petunjuk?				2	4	28	4.67

No.	Pertanyaan	STS	TS	RR	S	SS	Total Perhitungan	Nilai Rata - Rata
2.	Apakah proses pemilihan <i>tray</i> dan nomor meja dapat dilakukan dengan mudah dan tanpa kebingungan?				1	5	29	4.83
3.	Apakah aplikasi memberikan umpan balik ( <i>feedback</i> ) yang jelas setelah Anda menekan tombol “Start” pengantarannya?				2	4	28	4.67
4.	Apakah Anda merasa aplikasi cukup <i>responsif</i> (tidak lambat atau <i>freeze</i> ) selama digunakan?				3	3	27	4.50
5.	Apakah tampilan antarmuka (UI) aplikasi TIFA mudah dibaca dan tidak membingungkan?				4	2	26	.4.33
6.	Apakah Anda pernah mengalami <i>error</i> atau aplikasi tertutup sendiri ( <i>crash</i> ) saat melakukan <i>input</i> data?				4	2	26	4.33
7.	Apakah Anda merasa <i>input tray</i> dan nomor meja aman dari kesalahan (misalnya tidak bisa klik jika kosong)?				4	2	26	4.33
8.	Apakah Anda merasa fitur-fitur aplikasi ini membantu mempercepat proses pengiriman pesanan di jam sibuk?				3	3	27	4.50
9.	Sejauh ini, apakah Anda merasa puas dengan pengalaman penggunaan aplikasi TIFA?				2	4	28	4.67

#### D. Skenario Detil Web Monitoring

Bagian ini memuat daftar pertanyaan yang disusun untuk mengevaluasi pengalaman pengguna saat berinteraksi dengan web monitoring. Kuesioner dirancang menggunakan Skala Likert dengan nilai 1 sampai 5, guna mengukur tingkat kemudahan penggunaan, kejelasan antarmuka, dan efektivitas sistem dalam mendukung proses kerja administrator. Hasil dari

pengisian tabel ini akan digunakan sebagai dasar analisis dalam *User Acceptance Testing* (UAT).

**Tabel 5. 36 Skenario UAT Web Monitoring**

No.	Pertanyaan	Nilai				
		1	2	3	4	5
1.	Apakah Anda dapat dengan mudah <i>login</i> ke dalam sistem web monitoring menggunakan akun Anda?					
2.	Apakah informasi robot, log, dan grafik ditampilkan secara <i>real-time</i> tanpa perlu <i>refresh</i> manual?					
3.	Apakah informasi yang ditampilkan (informasi robot, log, dan grafik) mudah dipahami oleh Anda?					
4.	Apakah informasi robot, log, dan grafik yang ditampilkan di web monitoring sudah cukup lengkap untuk kebutuhan operasional Anda?					
5.	Apakah Anda menerima notifikasi terkait robot pada web monitoring?					
6.	Apakah navigasi antar halaman dalam web monitoring terasa cepat dan tidak membingungkan?					
7.	Apakah ukuran tampilan (layout) cocok untuk digunakan di berbagai perangkat seperti laptop, monitor, atau tablet?					
8.	Apakah Anda tidak mengalami kendala saat membuka halaman web monitoring?					
9.	Apakah web monitoring memungkinkan Anda melacak histori pengantaran sebelumnya dengan mudah?					
10.	Sejauh ini, apakah Anda merasa web monitoring bermanfaat untuk mengawasi aktivitas robot TIFA di Tel-U Coffee?					

## E. Proses Pengujian Web Monitoring

Proses pengujian *User Acceptance Testing* (UAT) pada sistem web monitoring dilakukan untuk memvalidasi kesesuaian antara fitur yang dikembangkan dengan kebutuhan pengguna akhir, khususnya untuk Rifqy Fachrizi selaku administrator yang bertugas memantau aktivitas robot. Pengujian ini dilakukan secara langsung di lingkungan operasional Tel-U Coffee dengan

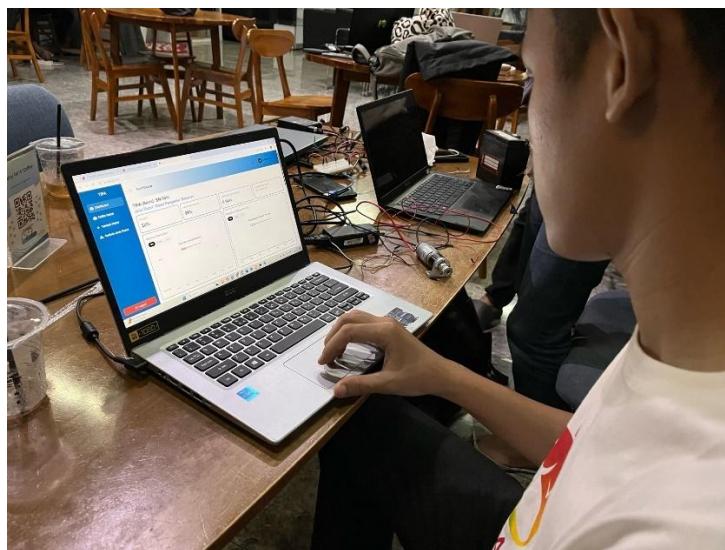
pendekatan berbasis observasi penggunaan dan pengisian kuesioner evaluasi. Berikut tahapan prosesnya:

1. Persiapan

- Administrator diberikan akses akun ke sistem web monitoring.
- Sistem *backend* dan koneksi internet diuji terlebih dahulu untuk memastikan stabilitas akses.
- Kuesioner UAT disiapkan dalam bentuk digital untuk diisi oleh pengguna.

2. Pelaksanaan Pengujian

- Administrator diminta untuk menjalankan beberapa fungsi utama seperti *login*, memantau robot, melihat grafik, serta menggunakan fitur navigasi dan notifikasi.
- Pengguna diminta memberikan penilaian secara subjektif terhadap aspek kemudahan penggunaan, kejelasan informasi, dan kelengkapan fitur.
- Semua interaksi dilakukan secara mandiri tanpa bantuan dari tim pengembang untuk mensimulasikan kondisi penggunaan riil.



**Gambar 5. 2 Proses Pengujian oleh Administrator**

3. Pengisian Kuesioner

- Setelah mencoba sistem, administrator mengisi kuesioner berdasarkan pengalaman selama penggunaan.
- Penilaian dilakukan dengan Skala Likert 1–5, dengan 10 pertanyaan yang telah disusun sebelumnya.

#### 4. Analisis Data

- Jawaban kuesioner dikonversi ke dalam bobot nilai.
- Nilai rata-rata dan persentase dihitung untuk mengevaluasi tingkat penerimaan sistem oleh pengguna.

#### 5. Tindak Lanjut

- Hasil pengujian dianalisis untuk mengetahui fitur mana yang sudah sesuai harapan dan mana yang perlu ditingkatkan.
- Temuan ini digunakan sebagai dasar untuk pengembangan sistem tahap selanjutnya.

### F. Hasil Pengujian Web Monitoring

*User Acceptance Testing* (UAT) dilakukan untuk memperoleh penilaian dari pengguna terkait kemudahan, kejelasan, dan *respons* sistem dalam web monitoring. Berdasarkan tanggapan pengguna terhadap sejumlah pertanyaan, diperoleh hasil yang sangat positif. Hal ini menunjukkan bahwa web monitoring telah mampu memberikan pengalaman pengguna yang baik dan sesuai harapan. Rincian nilai dapat dilihat pada tabel berikut.

**Tabel 5. 37 Hasil UAT Web Monitoring**

No.	Pertanyaan	STS	TS	RR	S	SS	Total Perhitungan	Nilai Rata Rata
1.	Apakah Anda dapat dengan mudah <i>login</i> ke dalam sistem web monitoring menggunakan akun Anda?				1		5	5
2.	Apakah informasi robot, log, dan grafik ditampilkan secara <i>real-time</i> tanpa perlu refresh manual?				1		5	5
3.	Apakah informasi yang ditampilkan (informasi robot, log, dan grafik) mudah dipahami oleh Anda?				1		5	5
4.	Apakah informasi robot, log, dan grafik yang ditampilkan di web monitoring sudah cukup lengkap untuk kebutuhan operasional Anda?				1		4	4
5.	Apakah Anda menerima notifikasi terkait robot pada web monitoring?				1		4	4

No.	Pertanyaan	STS	TS	RR	S	SS	Total Perhitungan	Nilai Rata Rata
6.	Apakah navigasi antar halaman dalam web monitoring terasa cepat dan tidak membingungkan?				1		5	5
7.	Apakah ukuran tampilan (layout) cocok untuk digunakan di berbagai perangkat seperti laptop, monitor, atau tablet?				1		5	5
8.	Apakah Anda tidak mengalami kendala saat membuka halaman web monitoring?				1		5	5
9.	Apakah web monitoring memungkinkan Anda melacak histori pengantaran sebelumnya dengan mudah?			1			4	4
10.	Sejauh ini, apakah Anda merasa web monitoring bermanfaat untuk mengawasi aktivitas robot TIFA di Tel-U Coffee?			1			4	4

### 5.3 Analisa Hasil Pengujian

Subbab ini membahas hasil analisis dari seluruh pengujian yang telah dilakukan terhadap sistem aplikasi TIFA dan web monitoring, baik dari sisi fungsi antarmuka maupun koneksi antar komponen internal seperti API. Analisis ini bertujuan untuk menilai sejauh mana sistem memenuhi spesifikasi teknis serta keandalan dalam menjalankan fitur-fitur utamanya.

#### 5.3.1 Security Testing

Hasil pengujian keamanan menunjukkan bahwa sistem mampu mengidentifikasi enam jenis kerentanan dengan tingkat risiko yang beragam, mulai dari risiko tinggi hingga informasi. Temuan paling kritis seperti akses metadata *cloud* secara publik dan absennya header keamanan penting seperti CSP dan X-Frame-Options menunjukkan bahwa pengujian memiliki tingkat keberhasilan yang cukup tinggi dalam menjawab permasalahan keamanan utama. Solusi yang diberikan seperti penambahan header keamanan dan pembatasan akses terhadap *endpoint* sensitif dinilai efektif secara teknis, meskipun membutuhkan validasi lanjutan pada lingkungan produksi.

Keberhasilan tersebut tidak lepas dari faktor pendukung utama seperti penggunaan alat otomatis OWASP ZAP yang handal dan efisien dalam mendeteksi kerentanan umum pada aplikasi web. Struktur aplikasi Juice Shop yang menyerupai sistem nyata juga memudahkan proses simulasi serangan dalam skenario yang *realistik*. Namun demikian, proses pengujian juga menghadapi sejumlah hambatan, antara lain keterbatasan dalam mendeteksi serangan yang bersifat kompleks dan kontekstual, serta keterbatasan pengujian di lingkungan yang tidak sepenuhnya mencerminkan kondisi sebenarnya di produksi.

Selain itu, solusi yang dihasilkan masih memiliki keterbatasan karena lebih bersifat reaktif terhadap temuan spesifik, bukan pendekatan preventif menyeluruh terhadap spektrum ancaman yang lebih luas. Oleh karena itu, rencana pengembangan berkelanjutan sangat diperlukan, seperti penerapan pengujian keamanan secara periodik di semua tahapan pengembangan, integrasi ke dalam *pipeline* CI/CD, serta pelatihan keamanan bagi pengembang agar aplikasi yang dibangun dapat mempertahankan standar keamanan yang tinggi dalam jangka panjang.

Salah satu kelebihan dari sistem web ini terletak pada penerapan keamanan penyimpanan kredensial pengguna melalui teknik *hashing password*. Sistem ini menggunakan algoritma Argon2id untuk mengenkripsi *password* saat proses registrasi. Argon2id merupakan algoritma *hashing* modern yang telah diakui secara internasional sebagai salah satu metode paling aman dan tahan terhadap berbagai jenis serangan, seperti brute-force, rainbow table, maupun side-channel attack.

Dengan menggunakan fungsi *password\_hash()* dan *password\_verify()* di sisi *backend*, sistem memastikan bahwa *password* pengguna tidak pernah disimpan dalam bentuk asli (*plaintext*) di dalam basis data. Hal ini memberikan perlindungan tambahan apabila terjadi kebocoran data, karena informasi yang diperoleh penyerang tetap tidak dapat digunakan tanpa mengetahui *input* asli dari *hash* tersebut.

Penerapan *hashing* ini menunjukkan bahwa sistem tidak hanya berfungsi secara fungsional, tetapi juga memperhatikan aspek keamanan data pengguna, yang merupakan bagian penting dalam pengembangan aplikasi web modern. Dengan pendekatan ini, sistem memberikan jaminan keamanan yang lebih kuat, khususnya dalam konteks pengelolaan autentikasi dan otorisasi berbasis token.

### 5.3.2 *Integration Testing*

Berdasarkan pengujian yang telah dilakukan terhadap berbagai *endpoint* API, diperoleh gambaran bahwa sistem mampu menangani alur komunikasi antar modul dengan baik. Setiap *endpoint* yang diuji berhasil menjalankan fungsinya secara tepat, mulai dari pengambilan data, pengiriman perintah, hingga pembaruan informasi dalam database. Hal ini menunjukkan bahwa integrasi antar bagian dalam sistem telah berjalan secara konsisten dan stabil.

#### A. Aplikasi

Pengujian integrasi *endpoint* API TIFA menunjukkan hasil yang sangat baik di semua skenario yang diuji. *Endpoint* GET /tes\_app/get\_images.php dan GET /tes\_app/get\_tables.php berhasil mengambil data peta dan semua nomor meja, memastikan informasi dapat diakses dengan benar. Dalam hal manajemen meja, *endpoint* POST /tes\_app/add\_table.php dan POST /tes\_app/delete\_table.php berfungsi dengan baik dalam menambahkan dan menghapus nomor meja dari database, menjaga konsistensi data. Pengambilan koordinat dan kondisi baterai melalui *endpoint* GET /tes\_app/get\_coordinates.php dan GET /tes\_app/get\_battery\_level.php juga berjalan lancar, memberikan data yang dibutuhkan untuk operasional robot.

Manajemen pesanan diuji melalui *endpoint* POST /tes\_app/insert\_order.php, GET /tes\_app/get\_orders.php, dan POST /tes\_app/delete\_order.php, yang ketiganya mampu memasukkan, mengambil, dan menghapus data pesanan secara akurat, termasuk nomor meja dan posisi *tray*. Pengendalian pintu robot melalui *endpoint* POST /tes\_app/door\_control.php berhasil dijalankan, memastikan sistem dapat mengontrol dan memantau kondisi pintu. *Endpoint* GET /tes\_app/deliver.php juga berhasil mengirim data pesanan ke robot, menunjukkan bahwa integrasi antara aplikasi dan perangkat keras telah berjalan mulus.

Hasil dari pengujian integrasi yang sudah sesuai antara hasil yang diharapkan dan hasil pengamatan dinyatakan berhasil. Semua fungsi sistem bekerja sesuai dengan apa yang diharapkan. Perhitungan akurasi pengujian dilakukan dengan rumus:

$$\text{Integration Testing} = \frac{\text{Total Berhasil}}{\text{Total Pengujian}} \times 100\%$$

$$\text{Integration Testing} = \frac{12}{12} \times 100\%$$

$$\text{Integration Testing} = 100\%$$

Berdasarkan hasil tersebut, seluruh fungsi dari aplikasi dinyatakan berjalan dengan baik dan sistem API TIFA siap digunakan secara langsung dalam lingkungan operasional.

## B. Web Monitoring

Pengujian integrasi *endpoint* API web monitoring menunjukkan hasil yang sangat baik di semua skenario yang diuji. *Endpoint* GET /api/robotData berhasil mengambil semua data robot, memastikan informasi perangkat dapat diakses dengan benar. Dalam hal autentikasi, *endpoint* POST /api/login berfungsi dengan baik, memungkinkan pengguna untuk *login* dengan sukses. Pengelolaan data robot juga berjalan lancar; *endpoint* POST /api/robotData dan POST /api/robotType mampu menambahkan data robot dan jenis robot baru ke dalam sistem, sedangkan *endpoint* PUT /api/robotData/:robotId berhasil memperbarui data robot yang ada. Selain itu, pengambilan data spesifik seperti peta yang tersimpan melalui GET /api/images, daftar jenis robot via GET /api/robotType, dan Detail robot berdasarkan ID dengan GET /api/robotData/:robotId semuanya berfungsi sesuai harapan, menampilkan data yang relevan tanpa kendala.

Hasil dari pengujian integrasi yang sudah sesuai antara hasil yang diharapkan dan hasil pengamatan dinyatakan berhasil. Semua fungsi sistem bekerja sesuai dengan apa yang diharapkan. Perhitungan akurasi pengujian dilakukan dengan rumus:

$$\begin{aligned} \text{Integration Testing} &= \frac{\text{Total Berhasil}}{\text{Total Pengujian}} \times 100\% \\ \text{Integration Testing} &= \frac{8}{8} \times 100\% \\ \text{Integration Testing} &= 100\% \end{aligned}$$

## C. Sensor Infrared

Pengujian integrasi sistem sensor infrared (IR) pada robot TIFA difokuskan untuk memastikan akurasi deteksi objek pada *tray* dan integrasi sensor IR dengan ESP32. Sistem berhasil merespons perubahan kondisi *tray* dengan tepat melalui kombinasi logika IR yang mengontrol buka-tutup pintu. Lima sensor IR ditempatkan pada setiap *tray* dan diuji dari kondisi penuh (semua IR = 1) hingga kosong (semua IR = 0) dengan parameter keputusan menggunakan total pembacaan IR ( $\Sigma$ IR), dimana:

1. Total pembacaan IR ( $\Sigma$ IR) digunakan sebagai parameter keputusan.
2. Bila  $\Sigma$ IR > 0, jika status *tray* berisi maka pintu terbuka.
3. Bila  $\Sigma$ IR = 0, jika status *tray* kosong maka pintu tertutup.

Semua kombinasi pengujian menghasilkan respon sesuai harapan tanpa *error* fungsional. Faktor pendukung keberhasilan dalam integrasi ini meliputi, logika pemrosesan yang

sederhana, efektif, dan kode ESP32 yang stabil. Namun demikian, pengujian masih memiliki keterbatasan yaitu hanya dilakukan pada kondisi cahaya standar (500–800 lux), belum mencakup gangguan sinyal, interferensi cahaya, atau *Noise listrik*, serta *delay respons* sensor ke *backend* tidak diukur.

Untuk pengembangan berkelanjutan, disarankan menambahkan timeout logging untuk mengukur *delay deteksi-respons*, menguji sensor pada kondisi lingkungan ekstrem, dan mengintegrasikan visualisasi status sensor di dashboard monitoring untuk memudahkan troubleshooting saat sistem beroperasi [59].

#### D. *Voltage sensor*

Pengujian *Voltage sensor* pada sistem TIFA dilakukan untuk memastikan akurasi pembacaan tegangan baterai serta kestabilan sistem monitoring melalui ESP32. Pengujian dilakukan dengan beberapa skenario, yaitu tegangan minimum (49.2V), tegangan maksimum (53V), tegangan menengah, dan pembacaan berulang untuk menguji konsistensi.

##### 1. Akurasi Pengukuran Tegangan

Sensor berhasil mengukur tegangan yang telah di-*step-down* dari baterai (hingga 53V) menjadi kurang dari 25V, kemudian dibaca oleh ESP32. Konversi nilai tegangan ke dalam bentuk persentase dilakukan menggunakan rumus proporsional berdasarkan selisih antara tegangan minimum dan maksimum.

##### 2. Konsistensi dan Repetisi

Setiap titik tegangan diuji sebanyak tiga kali, dan hasilnya konsisten. Hal ini menunjukkan

sensor, ADC internal ESP32, dan proses komunikasi data berfungsi stabil dan andal.

##### 3. Ketepatan ADC dan Korelasi Data

Nilai *raw* ADC (12-bit) menunjukkan korelasi yang baik dengan nilai tegangan dan persentase

yang dihitung.

##### 4. Validasi Logika Konversi

Logika penghitungan persentase tegangan terbukti valid dan andal:

- Tidak ditemukan *delay*, *spike*, atau *Noise*
- Pembacaan berlangsung *real-time* dan *responsif* terhadap perubahan nilai tegangan

#### 5.3.3 *Alpha Testing*

Analisis hasil pengujian alpha pada aplikasi TIFA dan web monitoring menunjukkan tingkat keberhasilan yang sangat tinggi dalam mengeksekusi setiap skenario yang telah

ditentukan. Dari pengujian yang mencakup berbagai fitur penting dalam sistem, seluruh aksi berhasil dijalankan tanpa ditemukan kendala atau kegagalan. Hal ini mengindikasikan bahwa sistem memiliki kestabilan dan keandalan dalam menjalankan fungsionalitas utamanya. Keberhasilan ini memperkuat keyakinan bahwa aplikasi TIFA dan web monitoring siap untuk melanjutkan ke tahap pengujian berikutnya dan digunakan oleh pengguna akhir secara efektif.

#### A. Aplikasi

Berdasarkan data pengujian yang dilakukan terhadap aplikasi TIFA, total terdapat 20 skenario pengujian yang mencakup fitur-fitur penting dalam sistem. Dari keseluruhan pengujian tersebut, 20 aksi berhasil dijalankan dengan baik, sementara tidak ada aksi yang mengalami kegagalan. Hal ini menunjukkan bahwa sistem mampu menangani setiap skenario sesuai dengan yang diharapkan. Perhitungan persentase keberhasilan dilakukan menggunakan rumus:

$$\text{Alpha Testing} = \frac{\text{Total Berhasil}}{\text{Total Pengujian}} \times 100\%$$

$$\text{Alpha Testing} = \frac{20}{20} \times 100\%$$

$$\text{Alpha Testing} = 100\%$$

Dari hasil tersebut dapat disimpulkan bahwa tingkat keberhasilan pengujian aplikasi TIFA mencapai 100%. Capaian ini menegaskan bahwa seluruh fungsi yang diuji dalam aplikasi berjalan dengan lancar dan sesuai spesifikasi, menjadikan sistem ini layak untuk digunakan dalam lingkungan operasional nyata. Keberhasilan ini didukung oleh Setiap *endpoint* API dirancang untuk menangani fungsi spesifik, seperti pengelolaan meja, pesanan, dan kontrol pintu, yang memudahkan pengujian. Selain itu, keterlibatan langsung tim pengembang memungkinkan respon cepat terhadap bug dan perbaikan saat pengujian dilakukan. Namun, keberhasilan ini juga memiliki keterbatasan: pengujian dilakukan hanya dalam lingkungan Tel-U Coffee oleh pengembang sendiri, tanpa melibatkan pengguna akhir, sehingga potensi bias cukup besar. Selain itu, pengujian hanya mencakup skenario normal tanpa melibatkan kondisi ekstrem seperti *network failure* atau data *corrupt*. Ke depan, pengembangan berkelanjutan perlu difokuskan pada uji stres sistem, peningkatan validasi *input*, serta penambahan mekanisme *Fallback* saat terjadi kesalahan, agar sistem siap menghadapi kondisi operasional nyata.

## B. Web Monitoring

Berdasarkan data pengujian yang dilakukan terhadap web monitoring, total terdapat 37 skenario pengujian yang mencakup fitur-fitur penting dalam sistem, termasuk halaman *login*, *Dashboard*, daftar robot, tambah robot, dan tambah jenis robot, serta komponen seperti header dan *sidebar*. Dari keseluruhan pengujian tersebut, 37 aksi berhasil dijalankan dengan baik, sementara tidak ada aksi yang mengalami kegagalan. Hal ini menunjukkan bahwa sistem mampu menangani setiap skenario sesuai dengan yang diharapkan. Perhitungan persentase keberhasilan dilakukan menggunakan rumus:

$$\text{Alpha Testing} = \frac{\text{Total Berhasil}}{\text{Total Pengujian}} \times 100\%$$

$$\text{Alpha Testing} = \frac{37}{37} \times 100\%$$

$$\text{Alpha Testing} = 100\%$$

Faktor pendukung keberhasilan di antaranya adalah implementasi UI yang *responsif*, keakuratan data yang sinkron dengan database, serta stabilitas API *backend* yang memungkinkan proses baca/tulis berjalan lancar. Tim pengembang memastikan koneksi jaringan dan *backend* aktif selama pengujian, sehingga tidak terjadi *timeout* atau *error* pada tampilan data. Meskipun seluruh skenario diuji dan berhasil, pengujian tidak mencakup skenario *multi-device access* atau *responsivitas* lintas *platform*. Untuk pengembangan lanjutan, sistem web perlu diuji lebih lanjut pada perangkat berbeda dan kondisi jaringan bervariasi, serta melengkapi seluruh menu yang belum aktif untuk memastikan bahwa sistem dapat digunakan oleh berbagai jenis pengguna di berbagai kondisi operasional.

### 5.3.4 User Acceptance Testing (UAT)

Analisis hasil pengujian *User Acceptance Testing* (UAT) terhadap sistem TIFA dilakukan untuk mengevaluasi sejauh mana sistem diterima oleh pengguna akhir. Berdasarkan data kuesioner yang dikumpulkan dan diolah menggunakan Skala Likert, diperoleh nilai rata-rata dari setiap pertanyaan yang mencerminkan persepsi dan pengalaman pengguna dalam menggunakan sistem. Nilai-nilai yang tinggi pada hasil UAT menunjukkan bahwa fitur-fitur utama sistem telah berhasil memenuhi ekspektasi pengguna dari segi kemudahan penggunaan, kejelasan tampilan, dan keefektifan fungsi. Dengan demikian, hasil UAT ini memberikan indikator positif terhadap kualitas sistem yang telah dikembangkan.

## A. Aplikasi

Untuk mendapatkan gambaran kuantitatif mengenai tingkat penerimaan pengguna terhadap sistem TIFA, dilakukan analisis hasil kuesioner menggunakan metode Skala Likert. Setiap pertanyaan dalam kuesioner diberi bobot nilai mulai dari 1 (Sangat Tidak Setuju) hingga 5 (Sangat Setuju). Nilai-nilai ini kemudian diolah menjadi rata-rata untuk setiap pertanyaan.

$$Nilai \ Rata - Rata = \frac{\Sigma(\text{Bobot} \times \text{Frekuensi})}{\text{Total} \ \text{Responden}}$$

Setelah memperoleh nilai rata-rata dari masing-masing pertanyaan, dilakukan perhitungan persentase untuk mengetahui sejauh mana sistem memenuhi ekspektasi pengguna. Persentase ini dibandingkan dengan nilai maksimum Skala Likert (yaitu 5), sehingga dapat menggambarkan derajat kepuasan atau penerimaan pengguna secara lebih terukur [60], [61], [62].

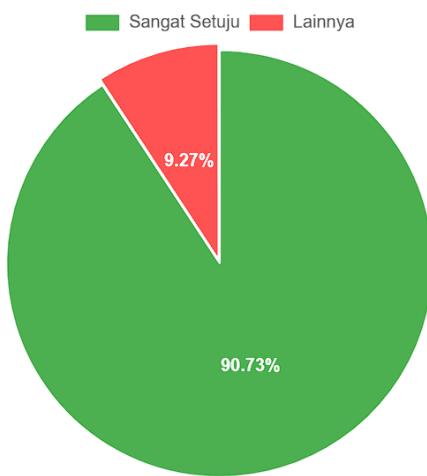
$$\text{Persentase} = \frac{\text{Nilai} \ Rata - Rata}{\text{Nilai} \ Maksimum} \times 100\%$$

**Tabel 5. 38 Analisa Hasil UAT Aplikasi**

No.	Pertanyaan	Persentase	Keterangan	Kesimpulan
1.	Apakah Anda dapat memahami fungsi dari setiap tombol pada halaman utama aplikasi TIFA tanpa bantuan petunjuk?	93.4	Sangat Setuju	Persentase Rata - Rata : 90.73 Kesimpulan : Sangat Setuju
2.	Apakah proses pemilihan <i>tray</i> dan nomor meja dapat dilakukan dengan mudah dan tanpa kebingungan?	96.6	Sangat Setuju	
3.	Apakah aplikasi memberikan umpan balik ( <i>feedback</i> ) yang jelas setelah Anda menekan tombol “Start” pengantar?	93.4	Sangat Setuju	
4.	Apakah Anda merasa aplikasi cukup <i>responsif</i> (tidak lambat atau <i>freeze</i> ) selama digunakan?	90,0	Sangat Setuju	
6.	Apakah Anda pernah mengalami <i>error</i> atau aplikasi tertutup sendiri ( <i>crash</i> ) saat melakukan <i>input</i> data?	86.6	Sangat Setuju	
7.	Apakah Anda merasa <i>input tray</i> dan nomor meja aman dari	86.6	Sangat Setuju	

No.	Pertanyaan	Persentase	Keterangan	Kesimpulan
	kesalahan (misalnya tidak bisa klik jika kosong)?			
8.	Apakah Anda merasa fitur-fitur aplikasi ini membantu mempercepat proses pengiriman pesanan di jam sibuk?	90.0	Sangat Setuju	
9.	Sejauh ini, apakah Anda merasa puas dengan pengalaman penggunaan aplikasi TIFA?	93.4	Sangat Setuju	

Percentase Keseluruhan (90.73% vs 9.27%)



**Gambar 5. 3 Pie Chart Analisa UAT Aplikasi**

*Pie Chart* ini secara visual menunjukkan bahwa semua pertanyaan memiliki tingkat kepuasan yang sangat tinggi (di atas 86%), dengan rata-rata persentase keseluruhan sebesar 90.73%, sesuai dengan kesimpulan "Sangat Setuju". Faktor pendukung keberhasilan dalam pengujian UAT aplikasi adalah kesesuaian antara fungsionalitas aplikasi dengan kebutuhan pengguna di lapangan [63].

Adapun faktor penghambat dan keterbatasan solusi terletak pada lingkup pengujian yang masih terbatas. Pengujian hanya melibatkan barista tanpa melibatkan pelanggan sebagai salah satu pihak yang juga merasakan manfaat sistem. Selain itu, jumlah responden hanya enam orang, sehingga belum mencerminkan variasi preferensi pengguna yang lebih luas. Tidak terdapat variasi perangkat atau kondisi pengujian yang berbeda, misalnya uji di berbagai waktu operasional atau intensitas pesanan yang tinggi, sehingga sistem belum diuji ketahanannya dalam kondisi ekstrem.

## B. Web Monitoring

Untuk mendapatkan gambaran kuantitatif mengenai tingkat penerimaan pengguna terhadap sistem TIFA, dilakukan analisis hasil kuesioner menggunakan metode skala Likert. Setiap pertanyaan dalam kuesioner diberi bobot nilai mulai dari 1 (Sangat Tidak Setuju) hingga 5 (Sangat Setuju). Nilai-nilai ini kemudian diolah menjadi rata-rata untuk setiap pertanyaan.

$$Nilai\ Rata - Rata = \frac{\Sigma(\text{Bobot} \times \text{Frekuensi})}{\text{Total\ Responden}}$$

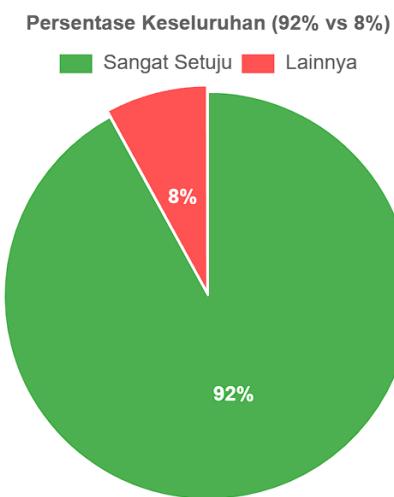
Setelah memperoleh nilai rata-rata dari masing-masing pertanyaan, dilakukan perhitungan persentase untuk mengetahui sejauh mana sistem memenuhi ekspektasi pengguna. Persentase ini dibandingkan dengan nilai maksimum skala Likert (yaitu 5), sehingga dapat menggambarkan derajat kepuasan atau penerimaan pengguna secara lebih terukur [60], [61], [62].

$$\text{Persentase} = \frac{\text{Nilai\ Rata} - \text{Rata}}{\text{Nilai\ Maksimum}} \times 100\%$$

**Tabel 5. 39 Analisa Hasil UAT Web Monitoring**

No.	Pertanyaan	Persentase	Keterangan	Kesimpulan
1.	Apakah Anda dapat dengan mudah <i>login</i> ke dalam sistem web monitoring menggunakan akun Anda?	100	Sangat Setuju	Persentasi Rata – Rata : 92 % Kesimpulan : Sangat Setuju
2.	Apakah informasi robot, log, dan grafik ditampilkan secara <i>real-time</i> tanpa perlu refresh manual?	100	Sangat Setuju	
3.	Apakah informasi yang ditampilkan (informasi robot, log, dan grafik) mudah dipahami oleh Anda?	100	Sangat Setuju	
4.	Apakah informasi robot, log, dan grafik yang ditampilkan di web monitoring sudah cukup lengkap untuk kebutuhan operasional Anda?	80	Setuju	
5.	Apakah Anda menerima notifikasi terkait robot pada web monitoring?	80	Setuju	
6.	Apakah navigasi antar halaman dalam web monitoring terasa	100	Sangat Setuju	

No.	Pertanyaan	Persentase	Keterangan	Kesimpulan
	cepat dan tidak membingungkan?			
7.	Apakah ukuran tampilan (layout) cocok untuk digunakan di berbagai perangkat seperti laptop, monitor, atau tablet?	100	Sangat Setuju	
8.	Apakah Anda tidak mengalami kendala saat membuka halaman web monitoring?	100	Sangat Setuju	
9.	Apakah web monitoring memungkinkan Anda melacak histori pengantaran sebelumnya dengan mudah?	80	Setuju	Persentasi Rata – Rata : 92 % Kesimpulan : Sangat Setuju
10.	Sejauh ini, apakah Anda merasa web monitoring bermanfaat untuk mengawasi aktivitas robot TIFA di Tel-U Coffee?	80	Setuju	



**Gambar 5. 4 Pie Chart Analisa UAT Web Monitoring**

*Pie Chart* ini secara visual menunjukkan bahwa semua pertanyaan memiliki tingkat kepuasan yang sangat tinggi (di atas 86%), dengan rata-rata persentase keseluruhan sebesar 92%, sesuai dengan kesimpulan "Sangat Setuju". Faktor pendukung dalam pengujian UAT web monitoring ditunjukkan oleh skor rata-rata yang tinggi dari satu orang administrator yang menjadi responden. Namun demikian, terdapat keterbatasan dari sisi jumlah responden yang hanya satu orang, sehingga cakupan evaluasi menjadi sangat sempit. Selain itu, tidak dilakukan pengujian oleh peran pengguna lain seperti supervisor, teknisi, atau pihak manajemen.

Pengujian juga tidak mencakup kondisi perangkat yang bervariasi atau skenario penggunaan simultan oleh banyak pengguna. Hal ini menjadi catatan bahwa meskipun hasil UAT menunjukkan tingkat penerimaan yang tinggi, validitasnya masih terbatas dan belum cukup untuk generalisasi.

#### 5.4 Kesimpulan

Berdasarkan implementasi Tel-U Interactive Food Assistant (TIFA) telah secara signifikan menjawab tantangan operasional Tel-U Coffee selama *peak hours*. Sistem ini terbukti efektif dalam meningkatkan efisiensi pelayanan melalui otomatisasi pengantaran dan pemantauan pesanan, dengan capaian kepuasan pengguna sebesar 90,73% untuk aplikasi dan 92% untuk web monitoring. Solusi ini juga berhasil menjawab keterbatasan jumlah staf serta peningkatan beban kerja saat lonjakan pesanan hingga 40%.

Berbagai keterbatasan pada sistem sebelumnya telah diatasi, termasuk peningkatan antarmuka melalui aplikasi Android yang lebih *responsif* dan integrasi web monitoring *real-time* untuk pengawasan terpusat. Arsitektur sistem yang menggabungkan aplikasi, web, database, dan robot telah menunjukkan keterpaduan fungsi dan kesiapan dalam mendukung operasional kompleks di industri *food service*.

Pengujian dilakukan secara menyeluruh melalui *Security Testing*, *Integration Testing*, *Alpha Testing*, dan *User Acceptance Testing* (UAT), dengan hasil seluruh fungsi utama bekerja 100% sesuai ekspektasi. Sistem sensor infrared juga terbukti mampu mendeteksi makanan dan minuman secara akurat. Namun, masih terdapat keterbatasan berupa jumlah responden terbatas serta belum diuji dalam skenario *multi-robot* atau terintegrasi dengan sistem POS.

Ke depan, pengembangan lanjutan diarahkan pada peningkatan skalabilitas dan keandalan sistem, termasuk pengujian ekstrem seperti gangguan jaringan, serta penambahan fitur seperti AI interaktif, dukungan *multi-robot*, dan mekanisme *fallback* untuk kondisi darurat.

Secara keseluruhan, sistem TIFA telah menunjukkan performa yang unggul dalam menjawab permasalahan inti dan memiliki potensi besar untuk terus dikembangkan. Kontribusinya dalam otomasi layanan *food service* dapat menjadi referensi penting bagi implementasi sistem serupa di masa mendatang.

## DAFTAR PUSTAKA

- [1] E. M. Manajemen, S. Bisnis, and D. Manajemen, “RESTAURANT XYZ PEAK HOUR WAITING TIME REDUCTION USING SIX SIGMA ANALYSIS,” *Journal of Social and Economics Research*, vol. 6, no. 1, 2024, [Online]. Available: <https://idm.or.id/JSER/index>.
- [2] I. Bernarto, J. Juliana, and A. Djakasaputra, “What Drives Customer Satisfaction? : Evidence From Customer Fast Food Restaurant Indonesia,” *Jurnal Aplikasi Bisnis dan Manajemen*, Sep. 2022, doi: 10.17358/jabm.8.3.711.
- [3] A. Pramezwary, D. Calista Oktaviani, F. E. Tania, and M. A. Benly, “UNDERSTANDING CUSTOMER LOYALTY IN INDONESIA QUICK SERVICE RESTAURANT INDUSTRY,” 2021.
- [4] V. Engesser, E. Rombaut, L. Vanhaverbeke, and P. Lebeau, “Autonomous Delivery Solutions for Last-Mile Logistics Operations: A Literature Review and Research Agenda,” Feb. 01, 2023, *MDPI*. doi: 10.3390/su15032774.
- [5] K. Shafique, B. A. Khawaja, F. Sabir, S. Qazi, and M. Mustaqim, “*Internet of Things* (IoT) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5G-IoT Scenarios,” 2020, *Institute of Electrical and Electronics Engineers Inc.* doi: 10.1109/ACCESS.2020.2970118.
- [6] M. Kalpana *et al.*, “Design and Implementation of Versatile Delivery Robot,” MDPI AG, Aug. 2024, p. 42. doi: 10.3390/engproc2024066042.
- [7] A. P. Priyadarshini, “The Impact of *User Interface Design* on *User Engagement*.” [Online]. Available: <http://www.ijert.org>
- [8] A. Hamm, “Future of Food Delivery: Drones, Robotics, and Automated Systems,” *African Journal of Food Science and Technology*, vol. 14, no. 12, pp. 1–02, 2023, doi: 10.14303//ajfst.
- [9] Copper Digital, “7 Benefits of Implementing IoT in Logistics.” Accessed: Oct. 26, 2024. [Online]. Available: <https://copperdigital.com/blog/benefits-implementing-iot-in-logistics/>

- [10] Fachrizi Rifqy, Novalisza Gebby, and Rafly Muhammad, “BUKU TUGAS AKHIR CAPSTONE DESIGN Mobilisasi Robot Pengantar Makanan Berbasis Odometry dan QR Detection,” Bandung, Dec. 2023.
- [11] D. Autor, “The journal of economic perspectives at 100 (issues),” 2012, *American Economic Association*. doi: 10.1257/jep.26.2.3.
- [12] PUDU, “BellaBot Robot Pengantar Kelas Premium.” Accessed: Nov. 23, 2024. [Online]. Available: <https://www.pudurobotics.com/id/product/Detail/bellabot>
- [13] Y. Chen, L. Chen, J. Ding, and Y. Liu, “Research on *Real-time* Obstacle Avoidance Motion Planning of Industrial Robotic Arm Based on Artificial Potential Field Method in Joint Space,” *Applied Sciences (Switzerland)*, vol. 13, no. 12, Jun. 2023, doi: 10.3390/app13126973.
- [14] N. Al Abdulhadi, M. S. Al Obaid, A. S. Al Fouzan, N. Al Hedaibi, P. Supervisor, and P. Khalid Sultan Professor Jibran Yousafzai, “Robot *Tray* System ELEG/CPEG 480-Capstone Design Project II Project Members,” 2021.
- [15] Figo, Jibran, and Nabila, “Hasil Wawancara dengan Pengguna,” Bandung, Nov. 2024. Accessed: Nov. 24, 2024. [Online]. Available: <https://drive.google.com/file/d/1rFFAXVdkDDLF4n2ov4WNLZsabkgKb4E-/view>
- [16] J. Wei, D. Shi, B. Yan, and Y. Hu, “A Large-scale Distribution and Deployment of Robot Task Based on MQTT Protocol and ROS,” 2017.
- [17] H. J. Sung and H. M. Jeon, “Untact: Customer’s acceptance intention toward robot barista in coffee shop,” *Sustainability (Switzerland)*, vol. 12, no. 20, pp. 1–16, Oct. 2020, doi: 10.3390/su12208598.
- [18] A. Saefullah, E. Sunandar, M. Nur Rifai, D. Jurusan Sistem Komputer STMIK Raharja, and A. STMIK Raharja Jurusan Sistem Komputer, “PROTOTIPE ROBOT PENGANTAR MAKANAN BERBASIS ARDUINO MEGA DENGAN INTERFACE WEB *BROWSER*,” 2017.
- [19] Agustin Electric Technology, “Robot Pengiriman Restoran.” Accessed: Nov. 24, 2024. [Online]. Available: <https://id.agustin-electric.com/robot/restaurant-robot/restaurant-delivery-robot.html>

- [20] B. A. Suren Sasin Rao, L. Sasidaran, and U. Tunku Abdul Rahman, “Simultaneous Localization and Mapping for Industrial Robots,” 2018.
- [21] LePage Pete and Andrew Rachel, “*Responsive web Design* basics.” Accessed: Nov. 24, 2024. [Online]. Available: <https://web.dev/articles/responsive-web-Design-basics>
- [22] Muallif, “Mekanisme Keamanan (Enkripsi, Autentikasi, Otorisasi).” Accessed: Nov. 24, 2024. [Online]. Available: <https://an-nur.ac.id/mekanisme-keamanan-enkripsi-autentikasi-otorisasi/>
- [23] F. Ilham Firmansyah, M. Syariffuddien Zuhrie, M. Rohman, and M. Aulia, “Sistem Obstacle Avoidance Pada Omnidirectional *Mobile* Robot Dengan Metode Artificial Potential Field (APF) Berbasis Fuzzy logic controller (FLC),” 2023.
- [24] Malwina Charko, “Step-by-Step Guide to Effective *User Acceptance Testing* for *Mobile* Apps.” Accessed: Nov. 23, 2024. [Online]. Available: [https://www.apptension.com/blog-posts/step-by-step-guide-to-effective-user-acceptance-Testing-for-mobile-apps?utm\\_source=chatgpt.com](https://www.apptension.com/blog-posts/step-by-step-guide-to-effective-user-acceptance-Testing-for-mobile-apps?utm_source=chatgpt.com)
- [25] D. D. Sari, M. Purba, and N. Umilizah, “IMPLEMENTASI *USER ACCEPTANCE TESTING* (UAT) PADA RANCANG BANGUN SISTEM INFORMASI PENERIMAAN PESERTA DIDIK BARU (PPDB),” *JCOSIS (Journal Computer Science and Information Syetem)*, [Online]. Available: <https://doi.org/10.61567>
- [26] D. Zeki Ablahd, B. Mohammed, A. Z. Ablahd, and B. Fadel Mohammed, “*Design* of a Remote Control of Home Automation using Android Studio.” [Online]. Available: [www.solidstatetechnology.us](http://www.solidstatetechnology.us)
- [27] H. Hussain, K. Khan, and F. Farooqui, “Comparative Study of Android Native and Flutter App Development,” 2021. [Online]. Available: <https://www.researchgate.net/publication/361208165>
- [28] R. F. Ramadhan and R. Mukhaiyar, “Penggunaan Database Mysql dengan Interface PhpMyAdmin sebagai Pengontrolan Smarthome Berbasis Raspberry Pi,” 2020.

- [29] C. Khawas and P. Shah, “Application of Firebase in Android App Development-A Study,” *Int J Comput Appl*, vol. 179, no. 46, pp. 49–53, Jun. 2018, doi: 10.5120/ijca2018917200.
- [30] M. FOTACHE and D. COGEAN, “NoSQL and SQL Databases for *Mobile* Applications. Case Study: MongoDB versus PostgreSQL,” *Informatica Economica*, vol. 17, no. 2/2013, pp. 41–58, Jun. 2013, doi: 10.12948/issn14531305/17.2.2013.04.
- [31] A. S. Ahmed, H. A. Marzog, and L. A. Abdul-Rahaim, “*Design* and implement of robotic arm and control of moving via IoT with Arduino ESP32,” *International Journal of Electrical and Computer Engineering*, vol. 11, no. 5, pp. 3924–3933, Oct. 2021, doi: 10.11591/ijece.v11i5.pp3924-3933.
- [32] “RemoteSensingIoTbasedAndroidControlledRobotTAJIM”.
- [33] Y. Irawan, Muhardi, R. Ordila, and R. Diandra, “Automatic floor cleaning robot using arduino and ultrasonic sensor,” *Journal of Robotics and Control (JRC)*, vol. 2, no. 4, pp. 240–243, Jul. 2021, doi: 10.18196/jrc.2485.
- [34] H. Shah, S. Zulfikar, and A. Bhutto, “Node.js Challenges in Implementation,” 2017. [Online]. Available: <https://www.researchgate.net/publication/318310544>
- [35] J. Vainikka, “Abstract Author Title Number of Pages Date,” 2018.
- [36] H. Suryotrisongko, D. P. Jayanto, and A. Tjahyanto, “*Design* and Development of *Backend* Application for Public Complaint Systems Using Microservice Spring Boot,” in *Procedia Computer Science*, Elsevier B.V., 2017, pp. 736–743. doi: 10.1016/j.procs.2017.12.212.
- [37] M. N. Mansor, N. A. A. Talib, S. A. Saidi, W. A. Mustafa, and N. F. Zamri, “Arduino IOT Based Inventory Management System Using Load Cell and NodeMCU,” *Journal of Advanced Research in Applied Sciences and Engineering Technology*, vol. 32, no. 3, pp. 12–25, Nov. 2023, doi: 10.37934/araset.32.3.1225.
- [38] Tarek Mohammad, “Using Ultrasonic and Infrared Sensors for Distance Measurement ,” 2009.
- [39] A. R. YEOLE, S. M. BRAMHANKAR, M. D. WANI, and M. P. MAHAJAN, “Smart Phone Controlled Robot Using ATMEGA328 Microcontroller,”

*International Journal of Innovative Research in Computer and Communication Engineering*, vol. 03, no. 01, pp. 352–356, Feb. 2015, doi: 10.15680/ijircce.2015.0301020.

- [40] G. Ersahin and H. Sedef, “Wireless Mobile Robot Control With Tablet Computer,” *Procedia Soc Behav Sci*, vol. 195, pp. 2874–2882, Jul. 2015, doi: 10.1016/j.sbspro.2015.06.411.
- [41] D. W. Nugraha, “PERANCANGAN SISTEM KONTROL ROBOT LENGAN YANG DIHUBUNGKAN DENGAN KOMPUTER.”
- [42] Levlin Mattias, “DOM benchmark comparison of the front-end JavaScript Frameworks React, Angular, Vue, and Svelte”.
- [43] K. Bielak, B. Borek, and M. Plechawska-Wójcik, “Web application performance analysis using Angular, React and Vue Frameworks,” 2022.
- [44] Amir Akbar Wicaksono, Yuli Kurnia Ningsih, and Indra Surjati, “Implementation of MQTT Protocol on ESP32-Based OEE Analysis Development Board,” *Emitor: Jurnal Teknik Elektro*, pp. 169–175, Aug. 2024, doi: 10.23917/emitor.v24i2.3908.
- [45] J. Lambert, R. Monahan, and K. Casey, “Power consumption profiling of a lightweight development board: Sensing with the INA219 and Teensy 4.0 microcontroller,” *Electronics (Switzerland)*, vol. 10, no. 7, Apr. 2021, doi: 10.3390/electronics10070775.
- [46] P. Jacko *et al.*, “Remote IoT Education Laboratory for Microcontrollers Based on the STM32 Chips,” *Sensors*, vol. 22, no. 4, Feb. 2022, doi: 10.3390/s22041440.
- [47] Q. Liang, Z. Wang, Y. Yin, W. Xiong, J. Zhang, and Z. Yang, “Autonomous aerial obstacle avoidance using LiDAR sensor fusion,” *PLoS One*, vol. 18, no. 6 JUNE, Jun. 2023, doi: 10.1371/journal.pone.0287177.
- [48] D. Wardhani, “KOMPARASI ANIMASI FORMAT LOTTIE, JSON, GIF DAN MP4 DENGAN METODE LOAD TESTING,” *Innovation and Technology*, vol. 1, no. 1, 2024.

- [49] D. Popa, I. Buciu, D. Popa, and I. Buciu, “CCC Publications Laravel and Vue.js as *tools* to control IoT *Devices* over the internet. Current *state-of-the-art*”, doi: 10.15837/ijccc.2025.3.7077.
- [50] N. Li and B. Zhang, “The Research on Single Page Application Front-end development Based on Vue,” in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Apr. 2021. doi: 10.1088/1742-6596/1883/1/012030.
- [51] K. Yank, “Building a Database-Driven Web Site Using PHP and MySQL.”
- [52] J. Chaitanya *et al.*, “INTERACTIVE WEB APPLICATION WITH CRUD USING PHP& MYSQL,” *International Journal of Information and Electronics Engineering*, vol. 15, no. 5, 2025, doi: 10.48047/ijiee.2025.15.5.60.
- [53] Z. Noori and C. Eriksson, “UI Performace Comparison of Jetpack Compose and XML in Native Android Applications.”
- [54] M. Albarka Umar, “Comprehensive study of *Software Testing*: Categories, levels, techniques, and types,” 2019. [Online]. Available: [www.IJARIIT.com](http://www.IJARIIT.com)
- [55] R. Zulkarnain, “SISTEM MONITORING MULTI SENSOR RUANG SERVER BERBASIS *INTERNET OF THINGS* (IOT) MENGGUNAKAN WEMOS D1 R2 (STUDI KASUS : PT LANCAR WIGUNA SEJAHTERA (LAWSON INDONESIA)),” 2024.
- [56] A. Yodi and R. P. Kristianto, “Analisis Evaluasi Aplikasi Kuis Anak Usia Dini melalui Manual dan Automation *Testing* Dengan teknik Black Box dan Espresso Analysis of Early Childhood Quiz Application Evaluation through Manual and Automation *Testing* Using Black Box and Espresso Techniques.”
- [57] I. Otaduy and O. Diaz, “*User Acceptance Testing* for Agile-developed web-based applications: Empowering customers through wikis and mind *maps*,” *Journal of Systems and Software*, vol. 133, pp. 212–229, Nov. 2017, doi: 10.1016/j.jss.2017.01.002.
- [58] N. S. R. Pillai and R. R. Hemamalini, “Hybrid *User Acceptance Test* Procedure to Improve the *Software Quality*,” *International Arab Journal of Information Technology*, vol. 19, no. 6, pp. 956–964, Nov. 2022, doi: 10.34028/iajit/19/6/14.

- [59] H. Nassima, “PEOPLE’S DEMOCRATIC REPUBLIC OF ALGERIA Thesis Presented for the Diploma of Academic Master Branch: Computer Science Option: Informatique System *Design* and Implementation of an Autonomous *Mobile Robot* Based on ESP-32.”
- [60] L. Damayanti, “Journal of Artificial Intelligence and Engineering Applications *Design* and Validation of a Web-Based E-CRM System for Baby Fashion Business: A Case Study of PT Rhinno Makmur Jaya Using Black Box and UAT *Testing*,” 2025. [Online]. Available: <https://ioinformatic.org/>
- [61] O. Tiomauli and O. Marleen, “Analysis of Booking Service Performace *Testing* on a Car Sales and Purchase Website Using the PIECES Method Based on the Implementation of *User Acceptance Testing* at a Car Service Booking Service Company,” *International Research Journal of Advanced Engineering and Science*, vol. 7, no. 1, pp. 119–126, 2022, [Online]. Available: [www.seva.id](http://www.seva.id)
- [62] S. Y. Chien, “A usability evaluation *Framework* for a *mobile* application in supporting home-based rehabilitation for stroke patients: A qualitative study,” *Digit Health*, vol. 11, Jan. 2025, doi: 10.1177/20552076251340183.
- [63] E. Arif and I. Paulina Soko, “The Evaluation of web-based and android face-to-face tutorial applications quality using the *User Acceptance Testing* (UAT) method,” *Journal of World Science*, vol. 1, no. 8, pp. 590–595, Aug. 2022, doi: 10.36418/jws.v1i8.76.

## LAMPIRAN



**Gambar 5. 5 Wawancara dengan Mitra**

### **Lampiran: Hasil Wawancara dengan Pengguna**

Topik Wawancara: Identifikasi Kebutuhan untuk Pengembangan Aplikasi Monitoring dan Robot Pengantar Makanan

Tanggal Wawancara: 19 November 2024

Lokasi Wawancara: Tel-U Coffee

Responden: Manajer Operasional – Pak Devan

Metode Wawancara: Tatap Muka

---

#### Pertanyaan dan Jawaban

1. Apakah Anda ingin aplikasi memiliki opsi untuk menyesuaikan fitur tertentu sesuai kebutuhan operasional kafe?

Jawaban:

- Perlu dikembangkan fitur yang memungkinkan robot berkeliling area kafe sambil mengambil pesanan tambahan dari pelanggan.

- Fitur pemesanan langsung melalui aplikasi perlu diperluas agar lebih terintegrasi dengan menu yang ada di kafe.

2. Apa saja informasi yang harus selalu ditampilkan di layar utama aplikasi? (*Contoh: lokasi robot, status baterai, status pesanan*)

Jawaban:

Lokasi robot, status baterai, dan status pesanan sangat penting untuk ditampilkan secara *real-time*.

3. Apakah Anda membutuhkan notifikasi otomatis untuk masalah tertentu? (*Contoh: baterai lemah, jalur terhalang, atau pesanan selesai*)

Jawaban:

- Sangat membutuhkan notifikasi otomatis untuk kondisi seperti kapasitas baterai rendah, jalur terhalang, atau pesanan selesai.
- Notifikasi yang memberikan solusi langsung akan sangat membantu, seperti notifikasi untuk mengubah rute jika jalur terhalang.

4. Apakah aplikasi perlu fitur untuk mengatur antrean pesanan jika robot sedang sibuk?

Jawaban:

- Ya, fitur antrean pesanan sangat penting untuk memastikan pelanggan tetap dilayani meskipun robot sedang menjalankan tugas lain.
- Antrean sebaiknya juga bisa dikelola dari perangkat lain seperti tablet kasir.

5. Apakah mode operasional robot (*standby, work*) perlu dapat diatur melalui aplikasi?

Jawaban:

Mode operasional seperti *standby* dan *work* perlu dapat diatur langsung melalui aplikasi agar fleksibel dan mudah menyesuaikan dengan kebutuhan kafe.

6. Apakah aplikasi perlu menyertakan panduan singkat atau bantuan interaktif untuk mempermudah penggunaan?

Jawaban:

- Panduan singkat atau bantuan interaktif sangat diperlukan, terutama untuk pengguna awam.

- Hal ini penting untuk menghindari kebingungan atau kesalahan operasional yang dapat menyebabkan malfungsi.

---

### Kesimpulan Hasil Wawancara

Berdasarkan hasil wawancara, terdapat beberapa poin penting yang perlu dipertimbangkan dalam pengembangan aplikasi dan sistem robot pengantar makanan: Fitur utama yang dibutuhkan meliputi notifikasi otomatis, pengelolaan antrean pesanan, dan pengaturan mode operasional untuk mendukung efisiensi layanan di kafe. Informasi yang wajib ditampilkan secara *real-time* di layar utama aplikasi meliputi lokasi robot, status baterai, dan status pesanan. Disarankan juga untuk menambahkan informasi seperti tingkat “stress” robot untuk memantau beban kerja. Panduan penggunaan atau bantuan interaktif sangat diperlukan untuk mendukung kemudahan operasional dan menghindari potensi kesalahan teknis.

Hasil wawancara ini menjadi dasar penting dalam merancang spesifikasi sistem yang sesuai dengan kebutuhan operasional Tel-U Coffee.

### Data Responden

#### Survey User Acceptance Testing Aplikasi T.I.F.A

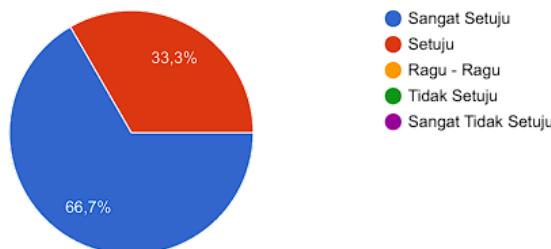
6 jawaban

[Publikasikan analytics](#)

Apakah Anda dapat memahami fungsi dari setiap tombol pada halaman utama aplikasi TIFA tanpa bantuan petunjuk?

Salin

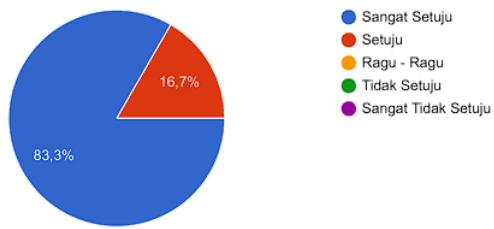
6 jawaban



 Salin

Apakah proses pemilihan tray dan nomor meja dapat dilakukan dengan mudah dan tanpa kebingungan?

6 jawaban

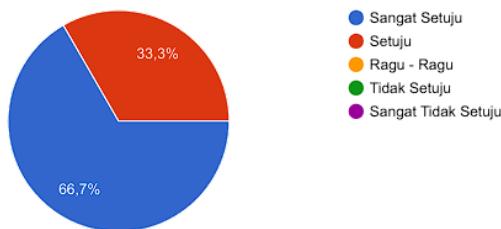


- Sangat Setuju
- Setuju
- Ragu - Ragu
- Tidak Setuju
- Sangat Tidak Setuju

 Salin

Apakah aplikasi memberikan umpan balik (feedback) yang jelas setelah Anda menekan tombol "Start" pengantarannya?

6 jawaban

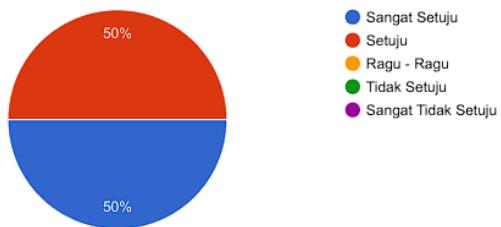


- Sangat Setuju
- Setuju
- Ragu - Ragu
- Tidak Setuju
- Sangat Tidak Setuju

 Salin

Apakah Anda merasa aplikasi cukup responsif (tidak lambat atau freeze) selama digunakan?

6 jawaban

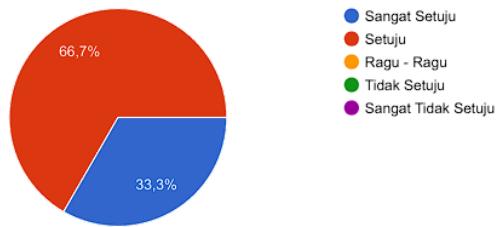


- Sangat Setuju
- Setuju
- Ragu - Ragu
- Tidak Setuju
- Sangat Tidak Setuju

 Salin

Apakah tampilan antarmuka (UI) aplikasi TIFA mudah dibaca dan tidak membingungkan?

6 jawaban

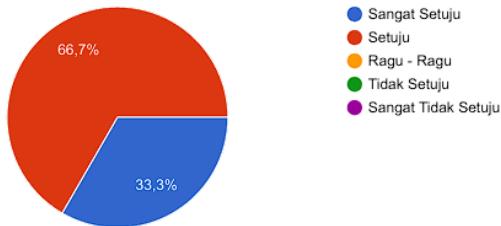


- Sangat Setuju
- Setuju
- Ragu - Ragu
- Tidak Setuju
- Sangat Tidak Setuju

 Salin

Apakah Anda pernah mengalami error atau aplikasi tertutup sendiri (crash) saat melakukan input data?

6 jawaban

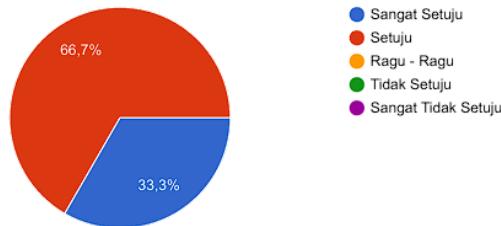


- Sangat Setuju
- Setuju
- Ragu - Ragu
- Tidak Setuju
- Sangat Tidak Setuju

 Salin

Apakah Anda merasa input tray dan nomor meja aman dari kesalahan (misalnya tidak bisa klik jika kosong)?

6 jawaban

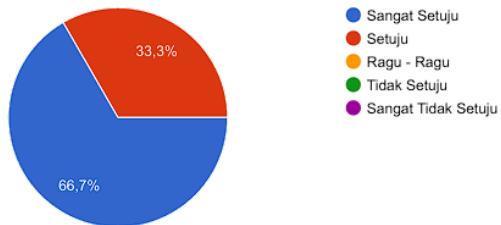


- Sangat Setuju
- Setuju
- Ragu - Ragu
- Tidak Setuju
- Sangat Tidak Setuju

 Salin

Apakah Anda merasa fitur-fitur aplikasi ini membantu mempercepat proses pengiriman pesanan di jam sibuk?

6 jawaban

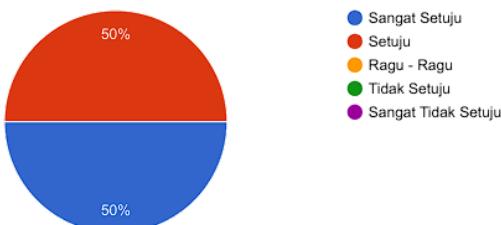


- Sangat Setuju
- Setuju
- Ragu - Ragu
- Tidak Setuju
- Sangat Tidak Setuju

 Salin

Sejauh ini, apakah Anda merasa puas dengan pengalaman penggunaan aplikasi TIFA?

6 jawaban

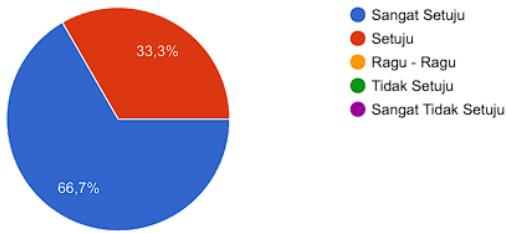


- Sangat Setuju
- Setuju
- Ragu - Ragu
- Tidak Setuju
- Sangat Tidak Setuju

Sejauh ini, apakah Anda merasa puas dengan pengalaman penggunaan aplikasi TIFA?

6 jawaban

 Salin



- Sangat Setuju
- Setuju
- Ragu - Ragu
- Tidak Setuju
- Sangat Tidak Setuju

## Survey User Acceptance Testing Web Monitoring

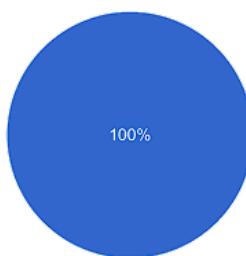
1 jawaban

[Publikasikan analytics](#)

Apakah Anda dapat dengan mudah login ke dalam sistem web monitoring menggunakan akun Anda?

1 jawaban

 Salin

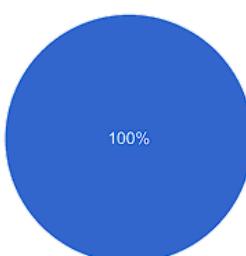


- Sangat Setuju
- Setuju
- Ragu - Ragu
- Tidak Setuju
- Sangat Tidak Setuju

Apakah informasi robot, log, dan grafik ditampilkan secara real-time tanpa perlu refresh manual?

1 jawaban

 Salin

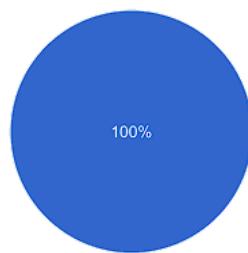


- Sangat Setuju
- Setuju
- Ragu - Ragu
- Tidak Setuju
- Sangat Tidak Setuju

Apakah informasi yang ditampilkan (informasi robot, log, dan grafik) mudah dipahami oleh Anda?

 Salin

1 jawaban

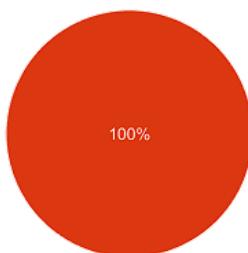


- Sangat Setuju
- Setuju
- Ragu - Ragu
- Tidak Setuju
- Sangat Tidak Setuju

Apakah informasi robot, log, dan grafik yang ditampilkan di web monitoring sudah cukup lengkap untuk kebutuhan operasional Anda?

 Salin

1 jawaban

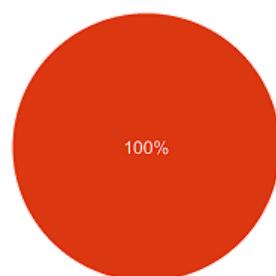


- Sangat Setuju
- Setuju
- Ragu - Ragu
- Tidak Setuju
- Sangat Tidak Setuju

Apakah Anda menerima notifikasi terkait robot pada web monitoring?

 Salin

1 jawaban

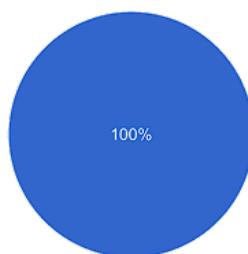


- Sangat Setuju
- Setuju
- Ragu - Ragu
- Tidak Setuju
- Sangat Tidak Setuju

Apakah navigasi antar halaman dalam web monitoring terasa cepat dan tidak membingungkan?

 Salin

1 jawaban

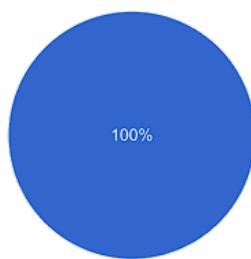


- Sangat Setuju
- Setuju
- Ragu - Ragu
- Tidak Setuju
- Sangat Tidak Setuju

Apakah ukuran tampilan (layout) cocok untuk digunakan di berbagai perangkat seperti laptop, monitor, atau tablet?

 Salin

1 jawaban

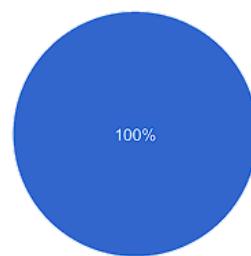


- Sangat Setuju
- Setuju
- Ragu - Ragu
- Tidak Setuju
- Sangat Tidak Setuju

Apakah Anda tidak pernah mengalami error saat membuka halaman web monitoring atau data tidak muncul?

 Salin

1 jawaban

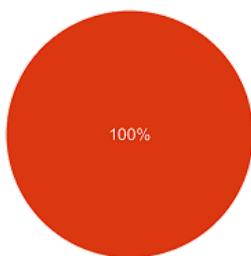


- Sangat Setuju
- Setuju
- Ragu - Ragu
- Tidak Setuju
- Sangat Tidak Setuju

Apakah web monitoring memungkinkan Anda melacak histori pengantaran sebelumnya dengan mudah?

 Salin

1 jawaban

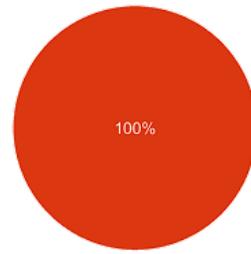


- Sangat Setuju
- Setuju
- Ragu - Ragu
- Tidak Setuju
- Sangat Tidak Setuju

Sejauh ini, apakah Anda merasa web monitoring bermanfaat untuk mengawasi aktivitas robot TIFA di Tel-U Coffee?

 Salin

1 jawaban



- Sangat Setuju
- Setuju
- Ragu - Ragu
- Tidak Setuju
- Sangat Tidak Setuju