

**PENGEMBANGAN APLIKASI MONITORING DAN
KONTROL OTOMATIS BUDIDAYA UDANG DENGAN
METODE *RAPID APPLICATION DEVELOPMENT (RAD)*
(Studi Kasus: Kelompok Budidaya Tambak Udang Sadewa Farm)**

TUGAS AKHIR

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

Oleh:

RIZKI ESA FADILLAH

121140084



**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI DAN INDUSTRI
INSTITUT TEKNOLOGI SUMATERA
LAMPUNG SELATAN
2025**

LEMBAR PENGESAHAN

Tugas Akhir dengan judul “Pengembangan Aplikasi Monitoring dan Kontrol Otomatis Budidaya Udang dengan Metode *Rapid Application Development* (RAD) (Studi Kasus: Kelompok Budidaya Tambak Udang Sadewa Farm)” adalah benar dibuat oleh saya sendiri dan belum pernah dibuat dan diserahkan sebelumnya, baik sebagian ataupun seluruhnya, baik oleh saya ataupun orang lain, baik di Institut Teknologi Sumatera maupun di institusi pendidikan lainnya.

Lampung Selatan, 10 Juni 2025
Penulis,

Rizki Esa Fadillah
NIM. 121140084

Diperiksa dan disetujui oleh,
Pembimbing

1. Aidil Afriansyah, S.Kom., M.Kom.
NIP.199104162019031015
2. Ilham Firman Ashari, S.Kom., M.T.
NIP.199303142019031018

Penguji

1. Eko Dwi Nugroho, S.Kom., M.Cs.
NIP.199102092024061001
2. Miranti Verdiana, S.Si., M.Si.
NIP.19920952022032008

Disahkan oleh,
Koordinator Program Studi Teknik Informatika
Fakultas Teknologi Industri
Institut Teknologi Sumatera

Andika Setiawan, S.Kom., M.Cs.
NIP. 199111272022031007

HALAMAN PERNYATAAN ORISINALITAS

Saya yang bertanda tangan di bawah ini

Menyatakan bahwa Tugas Akhir dengan judul : “PENGEMBANGAN APLIKASI MONITORING DAN KONTROL OTOMATIS BUDIDAYA UDANG DENGAN METODE *RAPID APPLICATION DEVELOPMENT (RAD)* (Studi Kasus: Kelompok Budidaya Tambak Udang Sadewa Farm)” adalah benar karya saya sendiri. Seluruh sumber baik yang dikutip maupun dirujuk dalam penulisan tugas akhir ini telah saya cantumkan secara benar sesuai dengan kaidah penulisan ilmiah.

Apabila di kemudian hari ditemukan adanya unsur plagiarisme dalam karya ini, maka saya bersedia menerima sanksi sesuai ketentuan yang berlaku di institusi saya.

Demikian pernyataan ini saya buat dengan sesungguhnya.

Lampung Selatan, 10 Juni 2025

Rizki Esa Fadillah
NIM. 121140084

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

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

Nama : Rizki Esa Fadillah
NIM : 121140084
Program Studi : Teknik Informatika
Fakultas : Fakultas Teknologi Industri
Jenis Karya : Tugas Akhir

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada **Institut Teknologi Sumatera** *Hak Bebas Royalti Noneksklusif* (*Non-exclusive Royalty Free Right*) atas karya ilmiah saya yang berjudul: “**PENGEMBANGAN APLIKASI MONITORING DAN KONTROL OTOMATIS BUDIDAYA UDANG DENGAN METODE RAPID APPLICATION DEVELOPMENT (RAD) (Studi Kasus: Kelompok Budidaya Tambak Udang Sadewa Farm)**” beserta perangkat yang ada (jika diperlukan). Dengan *Hak Bebas Royalti Noneksklusif* ini, Institut Teknologi Sumatera berhak untuk menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan Tugas Akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Lampung Selatan, 10 Juni 2025

Rizki Esa Fadillah
NIM. 121140084

KATA PENGANTAR

Dengan penuh rasa syukur dan hormat, penulis menyampaikan terima kasih yang sebesar-besarnya atas segala dukungan, bantuan, dan doa yang telah diberikan oleh berbagai pihak sehingga tugas akhir ini dapat diselesaikan dengan baik. Tugas akhir yang berjudul *"PENGEMBANGAN APLIKASI MONITORING DAN KONTROL OTOMATIS BUDIDAYA UDANG DENGAN METODE RAPID APPLICATION DEVELOPMENT (RAD) (Studi Kasus: Kelompok Budidaya Tambak Udang Sadewa Farm)"* ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana pada Program Studi Teknik Informatika, Institut Teknologi Sumatera.

Oleh karena itu, penulis menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Bapak Prof. Dr. I Nyoman Pugeg Aryantha, selaku rektor Institut Teknologi Sumatera.
2. Bapak Hadi Teguh Yudistira, S.T., Ph.D. selaku dekan Fakultas Teknologi Industri.
3. Bapak Andika Setiawan, S.Kom., M.Cs. selaku koordinator prodi Teknik Informatika.
4. Bapak Ilham Firman Ashari, S.Kom., M.T. selaku koordinator tugas akhir.
5. Bapak Aidil Afriansyah, S.Kom., M.Kom. selaku dosen pembimbing utama dan Bapak Ilham Firman Ashari, S.Kom., M.T. selaku dosen pembimbing kedua.

6. Bapak Eko Dwi Nugroho, S.Kom., M.Cs., Bapak Ir. Mugi Praseptiawan, S.T., M.Kom., dan Ibu Miranti Verdiana, S.Si., M.Si selaku dosen penguji.
7. Kedua Orang Tua, Bapak Marjo dan Ibu Supriyati, serta kakak tercinta yang selalu senatiasa memberikan dukungan serta doa terbaik kepada penulis.
8. Dimas Saputra selaku rekan tugas akhir *capstone*.
9. Kepada semua pihak yang tidak dapat saya sebutkan satu per satu, yang telah memberikan dukungan, dorongan, serta doa dengan tulus.

Akhir kata, penulis mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan, arahan, dan bantuan dalam penyelesaian Tugas Akhir ini. Penulis menyadari sepenuhnya bahwa tugas akhir ini masih jauh dari sempurna, baik dalam hal penyusunan maupun pemilihan kata. Penulis berharap tugas akhir ini dapat memberikan manfaat bagi serta menjadi referensi yang berguna bagi pihak-pihak yang membutuhkannya untuk pengembangan ke arah yang lebih baik. Semoga segala usaha dan kerja keras ini mendapatkan ridho dari Allah SWT.

RINGKASAN

Pengembangan Aplikasi Monitoring dan Kontrol Otomatis Budidaya Udang dengan Metode *Rapid Application Development (RAD)* (Studi Kasus: Kelompok Budidaya Tambak Udang Sadewa Farm)

Rizki Esa Fadillah

Budidaya udang merupakan sektor strategis yang berperan penting dalam perekonomian Indonesia, namun masih menghadapi tantangan, seperti pemantauan kualitas air kolam dan pemberian pakan yang masih dilakukan secara manual. Hal ini juga terjadi di Kelompok Budidaya Tambak Udang Sadewa Farm di Desa Pujodadi, Kecamatan Pardasuka, Kabupaten Pringsewu, Provinsi Lampung, dimana mereka masih menggunakan metode konvensional dalam pengelolaan tambak. Kualitas air seperti suhu, pH, salinitas, dan kekeruhan yang tidak terpantau dengan baik dapat menyebabkan udang mengalami stres. Selain itu, pemberian pakan yang tidak teratur dapat menyebabkan pertumbuhan udang yang kurang optimal. Oleh karena itu, dibutuhkan solusi teknologi untuk mendukung operasional budidaya udang.

Berdasarkan latar belakang tersebut, penelitian ini dirancang untuk mengembangkan aplikasi monitoring dan kontrol otomatis pada budidaya udang dengan metode *Rapid Application Development (RAD)*, sekaligus membantu petambak Sadewa Farm dalam memantau kualitas air secara *real-time* serta mengotomatisasi pemberian pakan dan *aerator*. Penelitian ini berfokus pada bagaimana pengembangan aplikasi monitoring dan kontrol otomatis berbasis *Internet of Things*

(*IoT*) yang terintegrasi dilakukan dan bagaimana kontribusinya terhadap kegiatan budaya, penelitian ini diharapkan memberikan solusi nyata yang dapat diterapkan langsung dalam operasional tambak, seperti memantau kualitas air secara *real-time* dan mengatur pemberian pakan serta *aerator* secara otomatis.

Aplikasi ini dikembangkan menggunakan metode *Rapid Application Development (RAD)*, yaitu pendekatan pengembangan perangkat lunak yang menekankan kecepatan, pembuatan prototipe secara iteratif, serta kolaborasi intensif antara pengembang dan pengguna. *RAD* memungkinkan penyesuaian cepat terhadap perubahan kebutuhan tanpa mengorbankan kualitas. Terdapat empat tahapan utama dalam *RAD*, yaitu *Requirements Planning* (Perencanaan Kebutuhan), *User Design* (Design Pengguna), *Construction* (Pengembangan Sistem), dan *Cutover* (Implementasi Sistem). Dalam penelitian ini, tahapan dimulai dari pengumpulan kebutuhan melalui wawancara dengan pemilik tambak, kemudian dilanjutkan dengan perancangan desain antarmuka yang diuji secara langsung kepada pengguna. Setelah desain disetujui, tahap Pengembangan dilakukan dengan membangun aplikasi menggunakan *Flutter* untuk antarmuka pengguna, *Express.js* sebagai *backend*, dan *MongoDB* sebagai penyimpanan data. Untuk mendukung pemantauan kualitas air secara *real-time*, aplikasi juga diintegrasikan dengan *Firebase Real-time Database*. Integrasi ini memungkinkan sistem secara otomatis memantau parameter seperti suhu, pH, salinitas, dan kekeruhan air, sekaligus mengontrol perangkat seperti *feeder* dan *aerator*.

Sebelum aplikasi diimplementasikan, aplikasi akan diuji untuk memastikan seluruh fitur aplikasi berfungsi dengan baik sesuai

kebutuhan. Pengujian pertama dilakukan dengan menggunakan *Black Box Testing* untuk mengevaluasi kinerja fungsionalitas aplikasi. Hasil pengujian menunjukkan bahwa seluruh skenario berhasil dijalankan tanpa kegagalan, dengan tingkat keberhasilan mencapai 100% yang menunjukkan bahwa aplikasi berfungsi dengan baik. Kemudian dilakukan pengujian untuk mengevaluasi aspek non-fungsional aplikasi menggunakan pengujian *System Usability Scale (SUS)* untuk mengukur tingkat kemudahan dan kepuasan pengguna terhadap aplikasi. Hasil pengujian menunjukkan skor rata-rata sebesar 82,83, yang termasuk dalam kategori “*Excellent*”. Nilai ini membuktikan bahwa aplikasi mudah digunakan dan dipahami oleh pengguna.

Untuk pengembangan ke depan, disarankan agar elemen *UI* ditinjau kembali guna meningkatkan kemudahan navigasi dan konsistensi desain. Selain itu, pengembangan dapat mempertimbangkan penggunaan teknologi alternatif seperti *React Native* atau *PostgreSQL* untuk performa yang lebih optimal. Fitur aplikasi juga dapat diperluas, seperti kustomisasi parameter budidaya, kontrol pompa sirkulasi, serta sistem pemberian pakan berbasis siklus pertumbuhan, agar aplikasi lebih fleksibel dan dapat digunakan pada berbagai jenis budidaya perairan lainnya.

ABSTRAK

Budidaya udang merupakan sektor strategis yang mendukung perekonomian Indonesia, namun masih menghadapi tantangan dalam pemantauan kualitas air dan pemberian pakan yang masih dilakukan secara manual. Hal ini juga terjadi di Sadewa Farm di Desa Pujodadi, Kecamatan Pardasuka, Kabupaten Pringsewu, Provinsi Lampung, Dimana mereka masih menggunakan metode konvensional dalam pengelolaan tambak. Penelitian ini bertujuan untuk mengembangkan aplikasi monitoring dan kontrol otomatis guna meningkatkan efisiensi pengelolaan tambak udang. Pengembangan aplikasi menggunakan metode *Rapid Application Development (RAD)* yang menekankan kecepatan, iterasi, dan keterlibatan pengguna secara intensif. Aplikasi dibangun menggunakan *Flutter* untuk antarmuka pengguna, *Express.js* sebagai *backend*, dan *MongoDB* sebagai penyimpanan data, serta terintegrasi dengan *Firebase Real-time Database* untuk pemantauan kualitas air secara *real-time* dan pengendalian perangkat seperti *feeder* dan *aerator* secara otomatis. Pengujian akan dilakukan menggunakan 2 metode, yaitu *Black Box Testing* untuk aspek fungsional dan *System Usability Scale (SUS)* untuk aspek non-fungsional. Hasil pengujian menunjukkan tingkat keberhasilan sistem sebesar 100% dan skor *SUS* sebesar 82,83 yang masuk dalam kategori “*Excellent*”. Hasil ini menunjukkan bahwa aplikasi tidak hanya berfungsi dengan baik, tetapi juga mudah digunakan dan dipahami oleh pengguna. Aplikasi ini diharapkan dapat menjadi solusi praktis dalam mendukung operasional budidaya udang.

Kata Kunci: Budidaya udang, *Internet of Things*, *Rapid Application Development*, Aplikasi Android

ABSTRACT

Shrimp farming is a strategic sector that supports the Indonesian economy, but still faces challenges in monitoring water quality and feeding which are still done manually. This also happens at Sadewa Farm in Pujodadi Village, Pardasuka District, Pringsewu Regency, Lampung Province, where they still use conventional methods in pond management. This study aims to develop an automatic monitoring and control application to improve the efficiency of shrimp pond management. Application development uses the Rapid Application Development (RAD) method which emphasizes speed, iteration, and intensive user involvement. The application is built using Flutter for the user interface, Express.js as the backend, and MongoDB as data storage, and is integrated with the Firebase Real-time Database for real-time water quality monitoring and automatic control of devices such as feeders and aerators. Testing will be carried out using 2 methods, namely Black Box Testing for functional aspects and System Usability Scale (SUS) for non-functional aspects. The test results show a system success rate of 100% and a SUS score of 82.83 which is included in the "Excellent" category. These results show that the application not only functions well, but is also easy to use and understand by users. This application is expected to be a practical solution in supporting shrimp farming operations.

Keywords: Shrimp farming, *Internet of Things*, Rapid Application Development, *Android Application*

DAFTAR ISI

LEMBAR PENGESAHAN.....	i
HALAMAN PERNYATAAN ORISINALITAS	iii
HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS	v
KATA PENGANTAR.....	vii
RINGKASAN	ix
ABSTRAK	xii
ABSTRACT	xiii
DAFTAR ISI	xiv
DAFTAR TABEL.....	xvii
DAFTAR GAMBAR.....	xviii
DAFTAR RUMUS	xxi
DAFTAR KODE	xxii
BAB I PENDAHULUAN	1
1.1 Latar Belakang Masalah	1
1.2 Rumusan Masalah	6
1.3 Tujuan Penelitian	7
1.4 Batasan Masalah	7
1.5 Manfaat Penelitian	8
1.6 Sistematika Penulisan.....	8
BAB II TINJAUAN PUSTAKA.....	11

2.1 Tinjauan Pustaka	11
2.2 Dasar Teori	24
2.2.1 Budidaya Udang.....	24
2.2.2 Kualitas Air Tambak Udang.....	24
2.2.3 Manajemen Pakan Udang	26
2.2.4 <i>Internet of Thinks (IoT)</i>	27
2.2.5 <i>Rapid Application Development (RAD)</i>	28
2.2.6 <i>Unified Modeling Language (UML)</i>	30
2.2.7 <i>Application Software</i>	36
2.2.8 <i>Flutter (Dart)</i>	36
2.2.9 <i>Express Js</i>	37
2.2.10 <i>Mongo DB</i>	38
2.2.11 <i>Firebase</i>	38
2.2.12 <i>Black Box Testing</i>	41
2.2.13 <i>System Usability Scale</i>	41
BAB III METODE PENELITIAN	45
3.1 Alur Penelitian	45
3.2 Penjabaran Langkah Penelitian	46
3.2.1 Identifikasi Masalah.....	46
3.2.2 Studi Literatur	46
3.2.3 Penerapan Metode <i>Rapid Application Development (RAD)</i>	47

3.2.4 Analisis dan Pembahasan	47
3.2.5 Kesimpulan dan Laporan	48
3.3 Alat dan Bahan Tugas akhir	48
3.3.1 Alat	48
3.3.2 Bahan	49
3.4 Metode Pengembangan	49
3.4.1 Perencanaan Kebutuhan (<i>Requirements Planning</i>).....	50
3.4.2 Desain Pengguna (<i>User Design</i>)	55
3.4.3 Pengembangan Sistem (<i>Construction</i>)	90
3.4.4 Implementasi (<i>Cutover</i>)	90
BAB IV HASIL DAN PEMBAHASAN	117
4.1. Hasil Penelitian	117
4.1.1. Desain Pengguna (Design <i>User</i>)	117
4.1.2. Pengembangan Sistem (<i>Construction</i>)	122
4.1.3. Implementasi (<i>Cutover</i>)	160
4.2. Analisis Hasil Penelitian	200
BAB V KESIMPULAN DAN SARAN	203
5.1. Kesimpulan	203
5.2. Saran	204
DAFTAR PUSTAKA	206
LAMPIRAN	214

DAFTAR TABEL

Tabel 2.1 Ringkasan Penelitian Terdahulu.....	13
Tabel 2.2 Simbol <i>Use case Diagram</i>	30
Tabel 2.3 Simbol <i>Activity Diagram</i>	32
Tabel 2.4 Simbol <i>Class Diagram</i>	33
Tabel 2.5 Simbol <i>Object Diagram</i>	35
Tabel 2.6 Skala Interpretasi Hasil <i>SUS</i>	44
Tabel 3.1 Hasil Wawancara	50
Tabel 3.2 Daftar Kebutuhan Fungsional	52
Tabel 3.3 Daftar Kebutuhan Non-Fungsional.....	54
Tabel 3.4 <i>Black Box Testing</i>	91
Tabel 3.5 Pertanyaan - Pertanyaan System <i>Usability Scale</i>	115
Tabel 4.1 Evaluasi Desain Iterasi Pertama.....	119
Tabel 4.2 Evaluasi Desain Iterasi Kedua	121
Tabel 4.3 Hasil Pengujian <i>Black Box Testing</i>	161
Tabel 4.4 Rincian Jawaban Skor <i>SUS</i> Responden 3	193
Tabel 4.5 Skor Asli Pengujian <i>SUS</i>	195
Tabel 4.6 Skor Hasil Pengujian <i>SUS</i>	198

DAFTAR GAMBAR

Gambar 2.1 Tahapan - Tahapan <i>RAD</i> [25].....	28
Gambar 2.2 Kategori dan Rentang Nilai Skor <i>SUS</i> [44].....	43
Gambar 3.1 Alur Metode Penelitian	45
Gambar 3.2 <i>Use case Diagram</i>	56
Gambar 3.3 <i>Activity Diagram Login</i>	58
Gambar 3.4 <i>Activity Diagram Logout</i>	59
Gambar 3.5 <i>Activity Diagram Lupa Password</i>	60
Gambar 3.6 <i>Activity Diagram Edit Profile</i>	61
Gambar 3.7 <i>Activity Diagram Manajemen User (Tambah User)</i>	62
Gambar 3.8 <i>Activity Diagram Manajemen User (Hapus User)</i>	63
Gambar 3.9 <i>Activity Diagram Manajemen User (Edit User)</i>	64
Gambar 3.10 <i>Activity Diagram</i> Manajemen Kolam (Tambah Kolam).....	65
Gambar 3.11 <i>Activity Diagram</i> Manajemen Kolam (Hapus Kolam)	66
Gambar 3.12 <i>Activity Diagram</i> Manajemen Kolam (Edit Kolam)....	67
Gambar 3.13 <i>Activity Diagram</i> Monitoring Kualitas Air.....	68
Gambar 3.14 <i>Activity Diagram</i> Mengatur Batasan Parameter Sensor.....	69
Gambar 3.15 <i>Activity Diagram</i> Melihat Informasi Perangkat Sensor.....	70
Gambar 3.16 <i>Activity Diagram</i> Notifikasi (Perubahan Batasan Parameter Sensor).....	71
Gambar 3.17 <i>Activity Diagram</i> Notifikasi (Kualitas Air)	72
Gambar 3.18 <i>Activity Diagram</i> Notifikasi (Pakan).....	73

Gambar 3.19 <i>Activity Diagram</i> Notifikasi (Jadwal dan Jumlah Pakan).....	74
Gambar 3.20 <i>Activity Diagram</i> Notifikasi (<i>Delay Aerator</i>).....	75
Gambar 3.21 <i>Activity Diagram</i> Melihat Riwayat Kualitas Air	76
Gambar 3.22 <i>Activity Diagram</i> Mengunduh Laporan Riwayat Kualitas Air	77
Gambar 3.23 <i>Activity Diagram</i> Kontrol Pakan Udang	78
Gambar 3.24 <i>Activity Diagram</i> Kontrol <i>Aerator</i>	79
Gambar 3.25 <i>Class Diagram</i>	80
Gambar 3.26 <i>Object Diagram</i>	82
Gambar 3.27 Halaman <i>Splash Screen</i> (Kiri) dan <i>Login</i> (Kanan)	84
Gambar 3.28 Halaman <i>Lupa Password</i> (Kiri) dan <i>Reset Password</i> (Kanan).....	84
Gambar 3.29 Halaman Beranda Admin (Kiri) dan <i>User</i> (Kanan).....	85
Gambar 3.30 <i>Popup</i> Menu Kolam (Kiri) dan Menu <i>Profile</i> (Kanan)	85
Gambar 3.31 Halaman <i>Profile</i> (Kiri) dan <i>Edit Profile</i> (Kanan)	86
Gambar 3.32 Halaman Tambah Kolam (Kiri) dan <i>Edit Kolam</i> (Kanan).....	87
Gambar 3.33 Halaman Manajemen <i>User</i> (Kiri), Tambah <i>User</i> (Tengah) dan <i>Edit User</i> (Kanan).....	87
Gambar 3.34 Halaman Monitoring (Kiri), Pengaturan Batasan Sensor (Tengah) dan Informasi Sensor (Kanan)	88
Gambar 3.35 Halaman Riwayat (Kiri) dan Laporan Monitoring (Kanan).....	89
Gambar 3.36 Halaman Notifikasi (Kiri) dan Kontrol (Kanan)	89
Gambar 4.1 Halaman Manajemen <i>User</i> Sebelum Perbaikan	118

Gambar 4.2 Halaman Manajemen <i>User</i> Setelah Perbaikan	119
Gambar 4.3 <i>Popup</i> Menu <i>Profile</i> Sebelum Perbaikan	120
Gambar 4.4 <i>Popup</i> Menu <i>Profile</i> Setelah Perbaikan	121
Gambar 4.5 Halaman <i>Splash Screen</i>	122
Gambar 4.6 Halaman <i>Login</i>	124
Gambar 4.7 Halaman Lupa <i>Password</i>	126
Gambar 4.8 Halaman <i>Reset password</i>	128
Gambar 4.9 Halaman Beranda Admin	130
Gambar 4.10 Halaman Beranda <i>User</i>	131
Gambar 4.11 <i>Popup</i> Menu Kolam.....	132
Gambar 4.12 <i>Popup</i> Menu <i>Profile</i>	134
Gambar 4.13 Halaman Tambah Kolam	136
Gambar 4.14 Halaman Edit Kolam	137
Gambar 4.15 Halaman <i>Profile</i>	139
Gambar 4.16 Halaman Edit <i>Profile</i>	140
Gambar 4.17 Halaman Manajemen <i>User</i>	141
Gambar 4.18 <i>Popup</i> Menu <i>User</i>	142
Gambar 4.19 Halaman Tambah <i>User</i>	144
Gambar 4.20 Halaman Edit <i>User</i>	146
Gambar 4.21 Halaman Monitoring.....	147
Gambar 4.22 Halaman Pengaturan Batasan Sensor	149
Gambar 4.23 Halaman Informasi Sensor.....	151
Gambar 4.24 Halaman Riwayat Data Monitoring	152
Gambar 4.25 Halaman Laporan Data Monitoring	154
Gambar 4.26 Halaman <i>Notifikasi</i>	156
Gambar 4.27 Halaman Kontrol Pakan dan <i>Aerator</i>	158

DAFTAR RUMUS

Rumus 2.1 <i>Feed Conversion Ratio (FCR)</i>	27
Rumus 2.2 Skor Akhis <i>System Usability Scale (SUS)</i>	43

DAFTAR KODE

Kode 4. 1 Halaman <i>Splash Screen</i>	123
Kode 4. 2 Halaman <i>Login</i>	125
Kode 4. 3 Halaman Lupa <i>Password</i>	127
Kode 4. 4 Halaman <i>Reset password</i>	129
Kode 4. 5 Halaman Beranda Admin	131
Kode 4. 6 Halaman Beranda <i>User</i>	132
Kode 4. 7 <i>Popup</i> Menu Kolam	133
Kode 4. 8 <i>Popup</i> Menu <i>Profile</i>	135
Kode 4. 9 Halaman Tambah Kolam.....	137
Kode 4. 10 Halaman Edit Kolam.....	138
Kode 4. 11 Halaman <i>Profile</i>	139
Kode 4. 12 Halaman Edit <i>Profile</i>	140
Kode 4. 13 Halaman Manajemen <i>User</i>	142
Kode 4. 14 <i>Popup</i> Menu <i>User</i>	143
Kode 4. 15 Halaman Tambah <i>User</i>	145
Kode 4. 16 Halaman Edit <i>User</i>	146
Kode 4. 17 Halaman Monitoring	148
Kode 4. 18 Halaman Pengaturan Batasan Sensor	150
Kode 4. 19 Halaman Informasi Sensor	151
Kode 4. 20 Halaman Riwayat Data Monitoring	153
Kode 4. 21 Halaman Laporan Data Monitoring	155
Kode 4. 22 Halaman <i>Notifikasi</i>	157
Kode 4. 23 Halaman Kontrol Pakan dan <i>Aerator</i>	159

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Udang merupakan salah satu komoditas ekspor unggulan Indonesia yang perlu ditingkatkan dalam hal kualitas dan kuantitas. Di tengah upaya tersebut, Provinsi Lampung, dengan potensi sumber daya pesisir dan laut yang melimpah, menjadi lokasi strategis untuk mengembangkan kegiatan budidaya udang [1]. Budidaya udang merupakan salah satu industri perikanan yang penting dan strategis bagi perekonomian seiring meningkatnya kesadaran masyarakat akan pentingnya konsumsi protein hewani yang berkualitas. Tingginya kebutuhan udang menjadikan usaha budidaya udang sebagai sektor yang sangat potensial untuk dikembangkan. Namun, sektor ini juga menghadapi berbagai tantangan yang dapat mempengaruhi produktivitas usaha, terutama dalam hal pengelolaan operasional tambak. Dalam budidaya udang, proses operasional seperti pemberian pakan dan pemantauan kualitas air tambak, sering kali mengalami kendala yang dapat menyebabkan hasil produksi menurun [2]. Kelompok Budidaya Tambak Udang Sadewa Farm di Desa Pujodadi, Kecamatan Pardasuka, Kabupaten Pringsewu, Provinsi Lampung menghadapi kendala yang serupa dalam memaksimalkan produktivitas budidaya ikan dan udang pada tambak mereka.

Berdasarkan wawancara yang dilakukan dengan bapak Kusdianto selaku pemilik tambak yang tergabung dengan Kelompok Budidaya Tambak Udang Sadewa Farm, diketahui bahwa operasional Pengelolaan tambak masih dilakukan secara manual, dimana pemilik

tambak harus memantau kondisi tambak secara langsung tanpa mengetahui secara pasti kualitas air dalam tambak. Kualitas air, seperti suhu, salinitas, pH, dan kekeruhan air, sangat penting karena dapat mempengaruhi pertumbuhan dan kelangsungan hidup benih udang. Kondisi air yang tidak terpantau dengan baik dapat menyebabkan stres pada udang, yang berpotensi mengakibatkan kematian dan menurunkan hasil produksi [3]. Untuk memastikan pertumbuhan yang optimal, kualitas air harus dijaga dalam kondisi ideal. Suhu air yang optimal untuk budidaya udang biasanya berkisar antara 28-32°C, pH yang optimal di antara 7,5-8, salinitas yang optimal antara 15-25 PPT (*Part per Thousand*), dan kekeruhan yang optimal berkisar antara 15-40 NTU (*Nephelometric Turbidity Unit*). Dengan menjaga parameter-parameter ini, risiko stres pada udang dapat diminimalkan, dan hasil produksi pun dapat ditingkatkan [4].

Selain itu, dari wawancara tersebut juga diketahui bahwa proses pemberian pakan pada tambak udang sadewa farm masih dilakukan secara manual dengan cara menyebar pakan langsung ke dalam kolam menggunakan tangan. Metode ini memiliki kekurangan, terutama ketika petani udang lupa atau terlambat dalam memberikan pakan, yang dapat mengakibatkan ketidakteraturan dalam jadwal pemberian. Jika pemberian pakan tidak dilakukan secara teratur, dampaknya akan terlihat pada pertumbuhan udang yang kurang optimal [5]. Agar pertumbuhan udang lebih optimal, pemberian pakan sebaiknya dilakukan secara teratur sebanyak 4 kali sehari, yaitu pada pukul 07.00, 10.00, 13.00, dan 16.00. Dengan frekuensi ini, udang vaname dapat mencapai pertumbuhan berat yang optimal, serta kualitas air tambak akan terjaga [6].

Berdasarkan permasalahan tersebut, diperlukan solusi berupa penerapan teknologi untuk mempermudah pengelolaan tambak udang, terutama dalam monitoring kualitas air dan pemberian pakan. Penerapan teknologi *Internet of Things (IoT)* dalam tambak udang memungkinkan monitoring *real-time* terhadap parameter penting seperti suhu, pH, salinitas, dan kekeruhan air dapat dilakukan secara otomatis dan terintegrasi, sehingga pemilik tambak tidak perlu memantau kondisi tambak secara manual setiap saat [7]. Selain itu, penerapan teknologi *IoT* memungkinkan pemberian pakan dan pengaturan *aerator* dilakukan secara otomatis sesuai jadwal yang teratur dan jumlah pakan yang tepat, membantu menjaga konsistensi pemberian pakan dengan jadwal teratur dan jumlah yang sesuai kebutuhan, sehingga pertumbuhan udang menjadi lebih optimal [8].

Meskipun teknologi *IoT* menawarkan kemampuan monitoring dan kontrol yang canggih, masih diperlukan aplikasi yang dapat mengintegrasikan data monitoring dari berbagai sensor dan memberikan kontrol otomatis yang sesuai. Aplikasi ini sangat penting untuk menyatukan informasi dari berbagai perangkat sensor, memungkinkan pengelolaan tambak udang menjadi lebih efektif [9]. Aplikasi ini memungkinkan pemantauan kualitas air secara *real-time*, pengaturan batasan parameter seperti suhu, pH, salinitas, dan tingkat kekeruhan, serta memberikan notifikasi jika ada perubahan yang melewati ambang batas aman. Selain itu, aplikasi ini menyimpan data monitoring historis yang memudahkan petani dalam melacak dan menganalisis kualitas air dalam jangka waktu tertentu, sekaligus mendukung pengaturan jadwal pemberian pakan secara otomatis, termasuk waktu dan jumlah pakan yang sesuai. Dengan fitur-fitur ini,

petani tambak udang dapat memanfaatkan data *real-time* untuk membuat keputusan lebih cepat dan akurat, serta menjaga kondisi tambak yang optimal. Dengan demikian, penerapan aplikasi monitoring dan kontrol otomatis ini diharapkan mampu mengurangi risiko kesalahan yang sering terjadi dalam pengelolaan manual budidaya udang [7].

Metode *Rapid Application Development (RAD)* akan diterapkan dalam pengembangan aplikasi ini, dengan pendekatan yang menekankan kecepatan, iterasi melalui pembuatan prototipe, kolaborasi intensif dengan pengguna, dan fleksibilitas menghadapi perubahan [10]. Tahapan *RAD* dimulai dengan *requirement planning*, di mana pengembang akan berkolaborasi dengan petani tambak untuk merumuskan kebutuhan sistem secara jelas. Tahap berikutnya adalah *User Design*, pada tahap ini fokus terhadap perancangan antarmuka pengguna yang intuitif dan pengujian prototipe untuk mendapatkan umpan balik. Pada fase *Construction*, pengembang akan melakukan pengkodean dan perbaikan sistem aplikasi, memastikan integrasi komponen seperti pengumpulan data dari sensor *IoT* dan kontrol otomatis untuk pemberian pakan. Kemudian, pada tahap *Cutover*, aplikasi akan diimplementasikan setelah dilakukan pengujian akhir. Dengan pendekatan *RAD*, pengembangan aplikasi monitoring dan kontrol otomatis budidaya udang dapat dilakukan dengan cepat dengan tetap menjaga kualitas dan memastikan kepuasan pengguna [11].

Tahap pengujian, aplikasi monitoring dan kontrol otomatis budidaya udang akan melalui serangkaian evaluasi untuk memastikan sistem dapat bekerja dengan baik. Metode pengujian yang akan diterapkan mencakup *Black Box Testing* untuk mengevaluasi

fungsionalitas setiap fitur, memastikan bahwa semua fungsi aplikasi berjalan sesuai dengan spesifikasi yang ditentukan. Pengujian ini dilakukan tanpa mempertimbangkan struktur internal aplikasi, sehingga fokus pada *output* yang dihasilkan dari *input* yang diberikan oleh pengguna [12]. Selain itu, *System Usability Scale (SUS)* akan digunakan untuk menilai aspek non-fungsional, termasuk kemudahan penggunaan dan kepuasan pengguna melalui pertanyaan-pertanyaan yang diberikan. Melalui pengujian ini, diharapkan dapat diidentifikasi area yang memerlukan perbaikan dan mengumpulkan umpan balik yang berguna untuk meningkatkan pengalaman pengguna [13].

Beberapa penelitian sebelumnya dalam pengembangan aplikasi berbasis *IoT* menunjukkan relevansi dengan penelitian ini. Pada tahun 2020, Harry Pratama Ramadhan, Condro Kartiko, dan Agi Prasetiadi melakukan studi berjudul "Monitoring Kualitas Air Tambak Udang Menggunakan Metode Data *Logging*". Penelitian ini menghasilkan aplikasi monitoring kualitas air berbasis *IoT* yang menggunakan *NodeMCU ESP8266*, sensor *LDR*, dan *Dallas 18B20*. Data yang diperoleh disimpan di *Firebase* dan dapat diakses melalui aplikasi *Android*, memungkinkan pemantauan kualitas air tambak secara *real-time*. Aplikasi ini terbukti efektif dalam mendeteksi perubahan kualitas air, seperti suhu dan kekeruhan, yang sangat penting untuk pengelolaan tambak udang [14]. Pada tahun 2022, Dian Noviandi dan Partaonan Harahap melakukan studi berjudul "Rancang Bangun Teknologi *Embedded System* Pemberi Pakan Ikan Berbasis *Internet of Things*". Penelitian ini menghasilkan alat pemberi pakan ikan berbasis *IoT* yang menggunakan *WeMos D1* sebagai pusat kendali. Alat pemberi pakan tersebut dapat dikendalikan melalui aplikasi *smartphone* yang

terhubung ke jaringan *internet*. Sistem ini dikembangkan menggunakan metode *Rapid Application Development (RAD)*, memungkinkan pengembangan sistem yang cepat dan fleksibel terhadap perubahan. Sistem ini terbukti fleksibel dalam mengakomodasi pemberian pakan dan pemantauan stok pakan, yang mendukung operasional tambak ikan [15]. Penelitian-penelitian ini menunjukkan relevansi teknologi *IoT* dan metode *RAD* dalam membangun aplikasi yang mendukung pengelolaan tambak secara modern, yang menjadi dasar bagi penelitian ini.

Berdasarkan penelitian sebelumnya, peneliti akan mengembangkan aplikasi monitoring dan kontrol otomatis untuk budidaya udang menggunakan metode *Rapid Application Development (RAD)*. Metode *RAD* dipilih karena keunggulannya dalam mempercepat proses pengembangan aplikasi melalui pembuatan prototipe yang iteratif, kolaboratif, dan fleksibel terhadap perubahan. Studi kasus yang akan digunakan melibatkan budidaya udang di Kelompok Budidaya Tambak Udang Sadewa Farm, yang menghadapi tantangan dalam pemantauan kualitas air dan pemberian pakan. Aplikasi ini akan dirancang untuk mengintegrasikan data dari sensor *IoT* untuk memantau kualitas air secara *real-time*, sekaligus mengotomatisasi pemberian pakan dan *aerator*. Sehingga, aplikasi ini diharapkan mampu mendukung pengelolaan tambak udang.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan diatas, rumusan masalah penelitian ini adalah sebagai berikut:

1. Bagaimana merancang dan mengembangkan aplikasi monitoring dan kontrol otomatis untuk budidaya udang menggunakan metode *Rapid Application Development (RAD)*?
2. Bagaimana aplikasi monitoring dan kontrol otomatis berperan dalam membantu petambak Udang di Sadewa Farm memantau kualitas air, sekaligus mengotomatisasi pemberian pakan dan *aerator*?

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah dipaparkan, tujuan dari penelitian ini adalah sebagai berikut:

1. Mengembangkan aplikasi monitoring dan kontrol otomatis untuk budidaya udang dengan menggunakan metode *Rapid Application Development (RAD)*.
2. Membantu petambak Udang Sadewa Farm dalam memantau kualitas air secara *real-time*, sekaligus mengotomatisasi pemberian pakan dan *aerator* untuk mendukung pengelolaan tambak udang.

1.4 Batasan Masalah

Adapun Batasan masalah dari penelitian ini adalah sebagai berikut:

1. Penelitian ini hanya akan fokus pada pengembangan aplikasi *mobile* berbasis *IoT*, untuk memantau kualitas air serta mengontrol sistem pemberian pakan dan *aerator* secara otomatis pada budidaya udang, khususnya di Kelompok Budidaya Tambak Udang Sadewa.
2. Pengujian aplikasi untuk monitoring terbatas pada parameter kualitas air seperti suhu, pH, kekeruhan, dan salinitas.

3. Aplikasi hanya menerima dan memproses data yang dikirimkan oleh perangkat sensor dan instalasi *Internet of Things (IoT)* yang telah terintegrasi dengan aplikasi.
4. Aplikasi *mobile* akan dikembangkan untuk sistem operasi *Android*, khususnya untuk versi *Android 11* atau yang lebih baru.
5. Pengujian aplikasi untuk monitoring menggunakan sensor *IoT* tertentu yang meliputi sensor suhu *DS18B20*, sensor pH *SEN0161*, dan sensor *turbidity analog* dari *Gravity* (kekeruhan).

1.5 Manfaat Penelitian

Adapun manfaat penelitian ini adalah dengan diimplementasikannya aplikasi monitoring dan kontrol otomatis budidaya udang dapat memberikan solusi teknologi bagi petambak udang untuk mengurangi risiko kesalahan dalam pemantauan kualitas air serta mengontrol pemberian pakan otomatis dan *aerator*.

1.6 Sistematika Penulisan

Sistematika yang digunakan pada penelitian ini dibagi menjadi beberapa bab, yaitu:

1. Bab I Pendahuluan

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

2. Bab II Tinjauan Pustaka

Bab ini berisi tinjauan pustaka dan dasar teori yang mendukung penelitian tugas akhir ini.

3. Bab III Metode Penelitian

Bab ini menjelaskan tentang alur penelitian, penjabaran langkah penelitian, alat dan bahan, serta metode pengembangan.

4. Bab IV Hasil dan Pembahasan

Bab ini menjelaskan tentang hasil penelitian dari rancangan penelitian beserta pengujian dan analisis hasilnya.

5. Bab V Kesimpulan dan Saran

Bab ini berisi Kesimpulan yang menjelaskan mengenai temuan utama penelitian, dan saran yang berisi rekomendasi untuk pengembangan lebih lanjut dari aplikasi ini.

BAB II

TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Pada penelitian ini, peneliti mendapatkan tinjauan pustaka yang berasal dari penelitian-penelitian sebelumnya yang berkaitan dengan topik penelitian. Penelitian-penelitian tersebut mencakup studi yang dilakukan oleh Harry Pratama Ramadhan, Condro Kaertiko, Agi Prasetyadi (2020) dalam penelitian berjudul Monitoring Kualitas Air Tambak Udang Menggunakan Metode Data *Logging* berhasil mengembangkan aplikasi monitoring kualitas air tambak udang berbasis *Android* yang dapat menampilkan data *real-time* dari perangkat data logger yang terhubung ke *Firebase* [14]. Penelitian yang dilakukan oleh Dian Noviandi dan Partaonan Harahap (2022) dengan penelitian Rancang Bangun Teknologi *Embedded System* Pemberi Pakan Ikan Berbasis *Internet of Things* menghasilkan alat *IoT* yang dapat memberikan pakan ikan, mengatur jadwal, dan memantau stok pakan secara *real-time* melalui aplikasi *mobile* [15]. Penelitian yang dilakukan oleh Sholeh Rachmatullah, Muhammad Yazir Zain, Anang Faktchur Rachman, Matsaini (2023) dalam penelitian Monitoring Kualitas Air Tambak Udang Vaname Berbasis *Internet of Things* berhasil mengimplementasikan perangkat *IoT* berbasis *website* yang memantau suhu air dan *TDS* secara *real-time* menggunakan *mikrokontroler ESP-32* [16]. Penelitian yang dilakukan oleh Raditia Dhamayanti, Glenn Chudra, Alfa Yohanis (2023) dengan penelitian Pengembangan Aplikasi Monitoring Tambak Ikan Berbasis *Internet of Things* mengembangkan aplikasi *IoT* yang memantau suhu, kadar

oksin, dan keasaman air tambak bandeng secara *real-time*, dengan pengujian lapangan yang menunjukkan hasil memuaskan [17]. Penelitian yang dilakukan oleh Joshua Renaldo Tape, Arthur Mourits Rumagit, Stanley D. S. Karouw (2022) melalui penelitian Rancang Bangun Aplikasi *Marketplace* Pakan Ternak berhasil membuat aplikasi *marketplace* yang mendukung proses jual beli pakan ternak secara efisien, dengan seluruh fungsi aplikasi berjalan baik berdasarkan pengujian [18]. Berikut ini merupakan ringkasan hasil penelitian terdahulu, yang dapat dilihat pada tabel 2.1 sebagai berikut:

Tabel 2.1 Ringkasan Penelitian Terdahulu

No.	Penulis [Tahun] [Judul]	Permasalahan Penelitian	Metode	Hasil Penelitian	Perbedaan
1.	Harry Pratama Ramadhan, Condro Kaertiko, Agi Prasetyadi [2020][Monito ring Kualitas Air Tambak Udang Menggunakan Metode Data <i>Logging</i>][14]	Banyak tambak udang yang mengalami kerugian karena kualitas air tidak terpantau dengan baik. Pengecekan kualitas air masih melalui laboratorium dan jarang dilakukan, menyebabkan	Data <i>Logging</i>	Aplikasi monitoring kualitas air tambak udang berhasil dikembangkan, hal ini dapat dibuktikan dari kemampuan sistem dalam mengirimkan data kualitas air yang terdeteksi oleh perangkat data logger, secara <i>real- time</i> ke <i>Firebase</i> . Data yang dikirimkan dapat diakses melalui aplikasi	1. Pada penelitian sebelumnya, aplikasi berfokus pada monitoring kualitas air. Sedangkan pada penelitian ini, aplikasi dapat melakukan monitoring kualitas air serta mengontrol pemberian pakan dan <i>aerator</i> .

No.	Penulis [Tahun] [Judul]	Permasalahan Penelitian	Metode	Hasil Penelitian	Perbedaan
		banyak udang yang sakit dan mati.		<i>Android</i> , yang menampilkan informasi terkini mengenai kondisi kualitas air tambak.	<p>2. Pada penelitian sebelumnya, aplikasi hanya memonitoring kualitas air berupa suhu dan kecerahan air. Sedangkan pada penelitian ini, aplikasi melakukan monitoring pada suhu, pH, salinitas, dan kekeruhan air.</p> <p>3. Pada penelitian sebelumnya,</p>

No.	Penulis [Tahun] [Judul]	Permasalahan Penelitian	Metode	Hasil Penelitian	Perbedaan
					penelitian berfokus pada alat dan aplikasi <i>android</i> , sementara pada penelitian ini berfokus pada aplikasi <i>android</i> saja.
2.	Dian Noviandi, Partaonan Harahap [2022] [Rancang Bangun]	Para peternak ikan air tawar mengalami penurunan pendapatan saat musim panen karena mereka harus membayar tenaga	<i>Rapid Application Developme nt (RAD)</i>	Alat pemberi pakan ikan berbasis <i>IoT</i> berhasil dikembangkan, hal ini dapat dibuktikan dari kemampuan alat <i>IoT</i> yang dapat dikendalikan melalui aplikasi <i>mobile</i>	1. Pada penelitian sebelumnya, berfokus pada budidaya ikan air tawar. Sedangkan pada penelitian ini, berfokus pada budidaya udang.

No.	Penulis [Tahun] [Judul]	Permasalahan Penelitian	Metode	Hasil Penelitian	Perbedaan
	Teknologi <i>Embedded System</i> Pemberi Pakan Ikan Berbasis <i>Internet of Things</i>][15]	kontrak yang hanya bertugas memberi makan ikan secara manual pada pagi dan sore hari.		untuk memberikan pakan, mengatur jadwal dan memantau stok pakan secara <i>real-time</i> .	2. Pada penelitian sebelumnya, aplikasi hanya dapat mengontrol pemberian pakan ikan secara otomatis. Sedangkan pada penelitian ini aplikasi dapat melakukan monitoring serta kontrol pakan dan <i>aerator</i> secara otomatis.

No.	Penulis [Tahun] [Judul]	Permasalahan Penelitian	Metode	Hasil Penelitian	Perbedaan
					3. Pada penelitian sebelumnya, penelitian berfokus pada alat dan aplikasi <i>android</i> , sementara pada penelitian ini berfokus pada aplikasi <i>android</i> saja.
3.	Sholeh Rachmatullah, Muhammad Yazir Zain, Anang	Pemantauan kualitas air tambak masih dilakukan dengan manual dan pelaporan dilakukan	<i>Waterfall</i>	Perangkat <i>IoT</i> berhasil diimplementasikan dalam bentuk <i>website</i> untuk monitoring kualitas air tambak udang vaname.	1. Pada penelitian sebelumnya, menggunakan metode <i>Waterfall</i> . Sedangkan pada penelitian ini,

No.	Penulis [Tahun] [Judul]	Permasalahan Penelitian	Metode	Hasil Penelitian	Perbedaan
	Fakthur Rachman, Matsaini [2023] [Monitoring Kualitas Air Tambak Udang Vaname Berbasis <i>Internet of Things</i>][16]	dalam bentuk catatan <i>excel</i> , sehingga menghambat pengambilan keputusan yang cepat dan tepat dalam pengelolaan tambak.		<i>Website</i> tersebut memungkinkan pemantauan suhu air dan <i>TDS</i> secara <i>real-time</i> , dengan data yang diperoleh dari sensor suhu <i>DS18B20</i> dan sensor <i>TDS</i> yang terhubung dengan mikrokontroler <i>ESP-32</i> .	menggunakan metode <i>Rapid Application Development</i> , sebagai metode pengembangannya. 2. Pada penelitian sebelumnya, platform yang dibuat berbasis <i>web</i> . Sedangkan pada penelitian ini, platform yang dibuat berbasis aplikasi <i>mobile</i> .

No.	Penulis [Tahun] [Judul]	Permasalahan Penelitian	Metode	Hasil Penelitian	Perbedaan
					3. Pada penelitian sebelumnya, penelitian berfokus pada alat dan aplikasi <i>android</i> , sementara pada penelitian ini berfokus pada aplikasi <i>android</i> saja.
4.	RADitia Dhamayanti, Glenny Chudra, Alfa Yohanis	Pengukuran kualitas air tambak ikan bandeng masih dilakukan secara tRADisional, yang	<i>Software Developme nt Lifecycle</i> (SDLC)	Aplikasi berbasis <i>Internet of Things (IoT)</i> berhasil dikembangkan menggunakan metode System Development Life	1. Pada penelitian sebelumnya, berfokus pada monitoring kualitas air. Sedangkan pada

No.	Penulis [Tahun] [Judul]	Permasalahan Penelitian	Metode	Hasil Penelitian	Perbedaan
	[2023][Penge mbangan aplikasi monitoring tambak ikan berbasis <i>Internet of Things</i>][17]	membutuhkan waktu dan tenaga lebih. Sehingga, kualitas air tambak yang tidak terjaga dapat berdampak negatif pada pertumbuhan dan kualitas ikan bandeng, yang sangat sensitif terhadap kondisi lingkungan.		Cycle (SDLC) untuk memantau suhu, kadar oksigen, dan keasaman air tambak secara <i>real-time</i> . Pengujian lapangan membuktikan bahwa aplikasi dan perangkat <i>IoT</i> berfungsi dengan baik, memberikan solusi yang efisien untuk pemeliharaan tambak bandeng. Meskipun terdapat tantangan seperti	penelitian ini, aplikasi dapat melakukan monitoring kualitas air serta mengontrol pemberian pakan dan <i>aerator</i> . 2. Pada penelitian sebelumnya, aplikasi memonitoring kualitas air berupa suhu, kadar oksigen dan tingkat keasaman air. Sedangkan pada

No.	Penulis [Tahun] [Judul]	Permasalahan Penelitian	Metode	Hasil Penelitian	Perbedaan
				<p>biaya investasi awal yang tinggi dan keterbatasan teknologi.</p>	<p>penelitian ini, aplikasi melakukan monitoring pada suhu, pH, salinitas, dan kekeruhan air.</p> <p>3. Pada penelitian sebelumnya, menggunakan metode <i>Software Development Lifecycle</i>. Sedangkan pada penelitian ini, menggunakan metode <i>Rapid Application</i></p>

No.	Penulis [Tahun] [Judul]	Permasalahan Penelitian	Metode	Hasil Penelitian	Perbedaan
					<i>Development</i> , sebagai metode pengembangannya.
5.	Joshua Renaldo Tape, Arthur Mourits Rumagit, Stanley D. S. Karouw [2022][Rancangan Bangun Aplikasi Marketplace	Proses jual beli pakan ternak selama ini masih dilakukan secara konvensional, di mana masyarakat harus keluar rumah untuk membeli produk pakan ternak. Selain itu, belum tersedia platform	<i>Rapid Application Development (RAD)</i>	Aplikasi <i>Marketplace</i> Pakan Ternak berhasil dikembangkan dan dapat diterapkan. Berdasarkan pengujian yang dilakukan, seluruh fungsi aplikasi beroperasi dengan baik, mendukung masyarakat untuk memesan produk pakan ternak secara	<p>1. Pada penelitian sebelumnya, berfokus membuat aplikasi marketplace. Sedangkan penelitian ini, berfokus membuat aplikasi monitoring dan kontrol otomatis budidaya udang.</p>

No.	Penulis [Tahun] [Judul]	Permasalahan Penelitian	Metode	Hasil Penelitian	Perbedaan
	Pakan Ternak][18]	digital yang memudahkan masyarakat untuk mendapatkan informasi produk pakan ternak secara praktis melalui <i>smartphone.</i>		efisien melalui platform digital.	2. Pada penelitian sebelumnya, tidak mencantumkan metode pengujian yang digunakan. Sedangkan penelitian ini, menggunakan metode pengujian <i>Black Box</i> dan <i>System Usability Scale (SUS)</i> .

2.2 Dasar Teori

Dalam melakukan penelitian ini diperlukan dasar acuan tentang teori dan konsep yang berhubungan dengan penelitian ini. Berikut dasar teori yang dijadikan referensi dalam penelitian ini.

2.2.1 Budidaya Udang

Akuakultur atau budidaya perikanan memiliki peran penting dalam penyediaan protein hewani, menjadikannya sektor yang sangat menjanjikan untuk pengembangan lebih lanjut. Di antara berbagai biota perairan yang dibudidayakan, udang merupakan salah satu komoditas unggulan dengan nilai ekonomi yang tinggi untuk dibudidayakan. Namun, sektor ini kerap kali mengalami tantangan dalam prosesnya, terutama terkait dengan risiko yang dapat menghambat keberhasilan budidaya. Beberapa risiko tersebut antara lain perubahan cuaca ekstrem, yang dapat mempengaruhi suhu dan salinitas air pada lingkungan tambak, serta serangan penyakit yang disebabkan stress dan kualitas air yang buruk dapat menurunkan kualitas dan kuantitas udang yang dihasilkan [19]. Sehingga, budidaya udang memerlukan perhatian penuh dalam setiap tahapnya, mulai dari pemberian pakan yang sesuai dengan kebutuhan nutrisi udang, hingga pemantauan dan pengelolaan kualitas air secara rutin untuk menjaga suhu, pH, kekeruhan, serta salinitas air, sehingga dapat mencegah timbulnya penyakit dan menghasilkan hasil yang optimal dalam budidaya udang [2].

2.2.2 Kualitas Air Tambak Udang

Kualitas air tambak merupakan faktor yang sangat penting dalam budidaya udang, karena kondisi air yang buruk dapat memengaruhi kesehatan dan pertumbuhan udang. Banyak petambak

udang mengalami kegagalan panen akibat perubahan cuaca yang tidak menentu, yang sering kali menyebabkan penurunan kualitas air tambak. Kondisi ini tidak hanya membuat udang rentan terhadap penyakit, tetapi juga dapat berujung pada kematian massal jika tidak segera diatasi. Oleh karena itu, untuk menghasilkan kualitas udang yang optimal, diperlukan lingkungan tambak yang mendukung pertumbuhan udang dengan baik, termasuk pemantauan berkala terhadap kualitas air tambak [20]. Ada empat parameter utama yang perlu dipantau dalam menjaga kualitas air tambak udang, yaitu suhu, pH, salinitas, dan kekeruhan yang akan dijelaskan sebagai berikut [4]:

1. Suhu

Udang tumbuh optimal pada suhu air antara 28°C hingga 32° dan memiliki toleransi antara 26°C hingga 35°C. Suhu yang terlalu rendah dapat menyebabkan penurunan aktivitas udang, sedangkan suhu yang terlalu tinggi dapat menyebabkan stres, mengganggu metabolisme, dan memperburuk kondisi kualitas air.

2. pH

pH adalah ukuran tingkat keasaman atau kebasaan air, yang mempengaruhi proses kimia dan biologi di dalam air. pH yang optimal untuk tambak udang berkisar antara 7,5 hingga 8,5. pH yang terlalu rendah atau tinggi dapat menyebabkan gangguan pada sistem pernapasan dan metabolisme udang, serta mengganggu keseimbangan biologi dalam tambak.

3. Salinitas

Salinitas mengukur kadar garam dalam air, yang berperan penting dalam keseimbangan osmosis udang. Salinitas yang optimal

untuk tambak udang berkisar antara 15 hingga 25 PPT. Salinitas yang terlalu tinggi atau rendah dapat mempengaruhi kelangsungan hidup udang.

4. Kekeruhan

Kekeruhan berhubungan dengan kejernihan air tambak. Kondisi kekeruhan optimal untuk tambak udang adalah antara 15-40 NTU. Kekeruhan yang tinggi dapat menghambat penetrasi cahaya ke dalam kolam, sehingga mengganggu *fotosintesis fitoplankton* yang penting untuk keseimbangan ekosistem dan produksi oksigen dalam air.

2.2.3 Manajemen Pakan Udang

Manajemen pakan adalah upaya untuk meningkatkan efisiensi pemanfaatan pakan dan meminimalkan sisa pakan yang terbuang di tambak. Pemberian pakan yang tidak tepat, seperti *underfeeding* atau *overfeeding*, dapat merusak kualitas air, menyebabkan stres pada udang, dan memperlambat pertumbuhannya. Selain itu, manajemen pakan yang buruk dapat menghasilkan sisa pakan yang mencemari air dan meningkatkan kadar amonia yang berbahaya bagi udang [21]. Untuk mengatasi masalah ini perlu adanya penjadwalan pemberian pakan yang teratur agar penggunaan pakan semakin efisien. Pemberian pakan yang baik sebaiknya dilakukan empat kali sehari pada pukul 07.00, 10.00, 13.00, dan 16.00, untuk menjaga kualitas air dan mendukung pertumbuhan optimal udang Vanamei [6].

Selain itu, dosis pakan harus disesuaikan dengan kebutuhan udang, dengan memperhatikan faktor seperti usia, ukuran, dan kondisi lingkungan, agar pemberian pakan dapat dimanfaatkan secara

maksimal. Pada fase awal, yaitu pada usia 1 - 25 hari, pemberian pakan dilakukan dengan teknik *blind feeding*, di mana pakan diberikan tanpa memperhatikan sisa pakan yang ada di tambak. Metode ini bertujuan untuk memastikan semua udang mendapatkan cukup pakan meskipun tidak ada pengukuran pasti terhadap sisa pakan yang ada di tambak. Setelah usia 26 hingga masa panen, penentuan pakan harian dilakukan secara lebih terstruktur dengan memperhitungkan kebutuhan pakan yang lebih spesifik, di mana *Feed Conversion Ratio (FCR)* menjadi indikator penting dalam manajemen pakan untuk mengukur efisiensi penggunaan pakan dalam mendukung pertumbuhan udang [21]. FCR dihitung dengan rumus:

$$FCR = \frac{\text{Jumlah Pakan (kg)}}{\text{Jumlah Berat Udang (kg)}} \quad (2.1)$$

FCR yang optimal berkisar pada rentang 1,1 hingga 1,2, nilai *FCR* yang semakin kecil menunjukkan bahwa pakan digunakan dengan baik untuk meningkatkan massa tubuh udang [22].

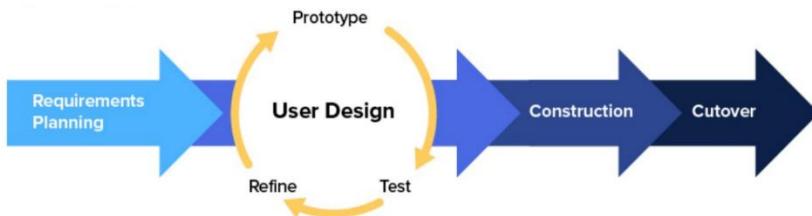
2.2.4 *Internet of Thinks (IoT)*

Internet of Things (IoT) merupakan salah satu teknologi yang sedang berkembang. Teknologi *IoT* hadir dalam berbagai bidang, memungkinkan peningkatan kualitas hidup dengan memudahkan akses ke informasi dan layanan tertentu. *IoT* merevolusi dunia dengan menghubungkan berbagai perangkat ke *internet*, sehingga menciptakan dunia yang lebih cerdas dan terintegrasi [23]. *IoT* beroperasi melalui tiga komponen utama dalam arsitekturnya, yaitu perangkat fisik yang telah dilengkapi dengan modul *IoT*, sarana konektivitas internet seperti *modem* atau *router nirkabel*, dan *cloud* sebagai pusat data untuk

menyimpan aplikasi serta *database*. IoT memanfaatkan pemrograman untuk menjalankan perintah secara otomatis, memungkinkan interaksi antara perangkat (*machine-to-machine*) tanpa keterlibatan langsung manusia dan tanpa batasan jarak. Internet berfungsi sebagai penghubung utama antara perangkat, sementara peran manusia terbatas pada pengaturan dan pemantauan perangkat secara langsung [24].

2.2.5 *Rapid Application Development (RAD)*

Rapid Application Development (RAD) merupakan pendekatan yang menekankan kecepatan, iterasi melalui pembuatan prototipe, kolaborasi intensif dengan pengguna, dan fleksibilitas menghadapi perubahan [10]. *RAD* memungkinkan proses pengembangan perangkat lunak yang lebih cepat tanpa menurunkan kualitas maupun kepuasan pengguna. Pendekatan ini juga mendorong kolaborasi yang lebih efektif dengan pengguna untuk memastikan bahwa produk yang dihasilkan sesuai dengan kebutuhan mereka. *RAD* juga mempermudah implementasi perubahan yang diperlukan selama pengembangan, dengan tujuan menghasilkan produk perangkat lunak yang lebih cepat namun tetap efektif [11]. Berikut ini adalah tahapan-tahapan dalam metode *RAD* yang dapat dilihat pada gambar 2.1 berikut.



Gambar 2.1 Tahapan - Tahapan *RAD* [25]

Pada gambar diatas terdapat 4 tahapan dalam pengembangan sistem menggunakan metode *RAD* [25], yaitu:

1. Perencanaan Kebutuhan (*Requirements Planning*)

Tahap ini untuk memahami kebutuhan pengguna dan mendefinisikan batasan sistem yang akan dibangun. Pada tahap ini akan mengumpulkan informasi yang diperlukan untuk merancang sistem yang sesuai dengan kebutuhan pengguna.

2. Desain Pengguna (*User Design*)

Pengguna dan pengembang bekerja bersama untuk membuat prototipe atau model dari sistem yang diinginkan, berdasarkan kebutuhan yang telah diidentifikasi. Pada tahap ini, berbagai diagram seperti *use case diagram*, *Activity Diagram*, *class diagram* dan *Object diagram* digunakan untuk menggambarkan bagaimana sistem akan berfungsi dan berinteraksi dengan pengguna.

3. Pengembangan Sistem (*Construction*)

Pengembang membangun fungsionalitas inti dari aplikasi dengan menggunakan alat pengembangan cepat dan *framework* yang memungkinkan integrasi cepat. Semua desain yang telah dibuat pada tahap sebelumnya akan dibangun menjadi sebuah aplikasi yang siap untuk diuji.

4. Implementasi (*Cutover*)

Tahap ini akan dilakukan pengujian sistem secara menyeluruh dan memastikan bahwa semua fungsionalitas bekerja sesuai dengan kebutuhan pengguna. Setelah sistem diuji dan divalidasi, implementasi dilakukan secara penuh di lingkungan produksi.

2.2.6 *Unified Modeling Language (UML)*

Unified Modeling Language (UML) adalah bahasa standar yang digunakan untuk memvisualisasikan, merancang, dan mendokumentasikan sistem perangkat lunak. *UML* menyediakan cara untuk membuat model aplikasi perangkat lunak yang dapat berjalan di berbagai platform dan ditulis dalam berbagai bahasa pemrograman. *UML* mendefinisikan notasi dan sintaks yang digunakan untuk menggambarkan berbagai *diagram* perangkat lunak, seperti *use case diagram*, *class diagram*, *statechart diagram*, *Activity Diagram*, *Object diagram*, *sequence diagram*, *collaboration diagram*, *component diagram*, dan *deployment diagram* [26]. Dalam penelitian ini, peneliti akan menggunakan *use-case diagram*, *activity diagram*, *class diagram* dan *Object diagram* untuk mengabstraksi sistem yang akan dikembangkan.

1. *Use-Case Diagram*

Use case Diagram adalah model untuk menggambarkan perilaku sistem yang akan dibangun dengan menunjukkan interaksi antara aktor dan sistem. *Diagram* ini digunakan untuk mengidentifikasi fungsi-fungsi dalam sistem serta pihak yang dapat mengakses fungsi tersebut [27]. Rincian simbol yang terdapat pada *diagram use-case* dan artinya dapat dilihat pada tabel 2.2 berikut.

Tabel 2.2 Simbol *Use case Diagram*

No	Nama	Simbol	Keterangan
1	<i>Use Case</i>		Merepresentasikan dialog satu dengan dialog lainnya

No	Nama	Simbol	Keterangan
2	<i>Actor</i>		Peran yang dimainkan oleh pengguna saat berinteraksi dengan <i>use case</i> .
3	<i>Extend</i>		Menunjukkan bahwa suatu case dapat tambahan fungsional dan bilamana fungsional tidak berfungsi maka sistem tetap bisa berjalan.
4	<i>Include</i>		Menunjukkan bahwa suatu case merupakan sebuah fungsionalitas.
5	<i>Association</i>		Apa yang menghubungkan antara objek satu dengan objek lainnya.
6	<i>System</i>		Menunjukkan data dalam paket yang menampilkan sistem secara terbatas.

2. Activity Diagram

Activity diagram menggambarkan aliran kerja atau aktivitas dalam suatu sistem, proses bisnis, atau menu dalam perangkat lunak, di mana aktivitas yang digambarkan adalah yang dilakukan oleh sistem. *Diagram* ini sering digunakan untuk mendefinisikan

rancangan proses bisnis, tampilan sistem, dan rancangan pengujian [28]. Rincian simbol yang terdapat pada pembuatan *activity diagram* dan artinya dapat dilihat pada tabel 2.3 berikut.

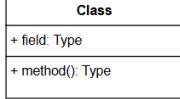
Tabel 2.3 Simbol *Activity Diagram*

No	Nama	Simbol	Keterangan
1	<i>Start</i>		Melambangkan status awal sebuah sistem.
2	<i>End</i>		Melambangkan status akhir sebuah sistem.
3	<i>Activity</i>		Melambangkan sebuah aktivitas yang dilakukan oleh sistem.
4	<i>Decision</i>		Melambangkan sebuah kondisi dimana terdapat pilihan aktivitas yang lebih dari satu.
5	<i>Swimlane</i>		Melambangkan pemisahan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi di dalamnya.

3. Class Diagram

Class Diagram adalah spesifikasi inti dalam desain berorientasi objek yang, ketika diinstansiasi, akan membentuk objek. *Class diagram* ini mendeskripsikan atribut (atau properti) dan menyediakan metode (atau fungsi) untuk memanipulasi keadaan tersebut. Setiap *class* umumnya terdiri dari tiga bagian utama: nama, atribut, dan metode [27]. Rincian simbol yang terdapat pada pembuatan *class diagram* dan artinya dapat dilihat pada tabel 2.4 berikut.

Tabel 2.4 Simbol *Class Diagram*

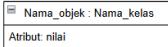
No	Nama	Simbol	Keterangan
1	<i>Dependency</i>-Use->	Menunjukkan operasi pada suatu <i>class</i> yang menggunakan <i>class</i> yang lain.
2	<i>Class</i>		<i>Class</i> adalah blok-blok pembangun pada pemrograman berorientasi objek. <i>Class</i> digambarkan sebagai sebuah kotak yang terbagi atas 3 bagian. Bagian atas adalah bagian nama dari <i>class</i> . Bagian tengah

No	Nama	Simbol	Keterangan
			mendefinisikan atribut <i>class</i> . Bagian akhir mendefinisikan metode-metode.
3	<i>Association</i>		Melambangkan hubungan antar objek.
4	<i>Composition</i>		Jika sebuah <i>class</i> tidak bisa berdiri sendiri dan harus merupakan bagian dari <i>class</i> yang lain, maka <i>class</i> tersebut memiliki relasi <i>Composition</i> terhadap <i>class</i> tempat dia bergantung tersebut.
5	<i>Generalization</i>		Menunjukkan hubungan antar kelas dengan umum (<i>general</i>) dengan kelas yang lebih spesifik.

4. Object Diagram

Object Diagram menggambarkan struktur sistem berdasarkan penamaan objek dan interaksi antar objek dalam sistem tersebut. *Object Diagram* memastikan bahwa setiap kelas yang telah didefinisikan dalam *diagram* kelas harus memiliki objek yang sesuai, karena tanpa objek yang relevan, pendefinisian kelas tersebut tidak dapat dibenarkan [29]. Rincian simbol yang terdapat pada pembuatan *class diagram* dan artinya dapat dilihat pada tabel 2.5 berikut.

Tabel 2.5 Simbol *Object Diagram*

No	Nama	Simbol	Keterangan
1	<i>Link</i>	_____	Menunjukkan hubungan antar objek-objek dalam sistem yang telah didefinisikan dalam <i>class diagram</i> .
2	Objek		Menyatakan entitas dalam sistem. Objek digambarkan sebagai sebuah kotak yang terbagi atas 2 bagian. Bagian atas adalah bagian nama dari objek dan nama <i>class</i> . Bagian bawah

No	Nama	Simbol	Keterangan
			mendefinisikan atribut yang dimiliki objek.

2.2.7 *Application Software*

Application Software atau yang sering disebut sebagai aplikasi, merupakan program independen yang dirancang untuk memenuhi kebutuhan bisnis secara spesifik. Aplikasi dapat memproses data bisnis atau teknis untuk mendukung operasi bisnis atau pengambilan keputusan manajerial/teknis. Selain untuk pemrosesan data, aplikasi juga digunakan untuk mengontrol fungsi bisnis secara langsung dan *real-time* [30]. Aplikasi dibagi menjadi tiga platform utama, yaitu *desktop*, *web*, dan *mobile*. Didorong oleh kemajuan pesat teknologi *smartphone* yang semakin canggih. Aplikasi *mobile* menjadi yang paling banyak digunakan [31], hal ini karena aplikasi *mobile* merupakan *software* yang dikembangkan khusus untuk perangkat *smartphone* menggunakan *J2ME (Java 2 Micro Edition)* yang merupakan platform yang menggunakan bahasa pemrograman *Java*, dengan komponen utama berupa *Java Virtual Machine (JVM)* yang memungkinkan program *Java* berjalan pada *emulator* atau perangkat genggam seperti *smartphone*, serta *Java API (Application Programming Interface)* [32].

2.2.8 *Flutter (Dart)*

Flutter adalah *framework open-source* yang dikembangkan oleh *Google* untuk pengembangan aplikasi *mobile* dengan satu basis kode yang bisa dijalankan pada *iOS* dan *Android*, yang membuatnya

lebih efisien dalam hal waktu dan biaya pengembangan. *Flutter* menyediakan berbagai *widget* untuk membangun antarmuka pengguna yang menarik dan responsif, mendukung aplikasi lintas platform pada perangkat *Android*, *iOS*, bahkan *desktop* seperti *Windows*, *Linux*, dan *macOS*. *Framework* ini juga menjadi solusi utama untuk aplikasi *Google Fuchsia* dan aplikasi berbasis *web* [33]. *Flutter* ditulis menggunakan bahasa *Dart*, sebuah bahasa pemrograman berorientasi objek (*OOP*) dengan sintaks yang mirip dengan *C++*, *Java*, dan *JavaScript*. *Dart* adalah bahasa yang dinamis dan memungkinkan kode untuk dijalankan langsung tanpa perlu dikompilasi terlebih dahulu, berkat dukungan dari *Dart VM*. Bahasa ini juga dapat dijalankan secara langsung pada *browser Chrome* tanpa kompilasi. *Dart*, dengan karakteristik yang mirip *Java* dan *JavaScript*, menjadi dasar bagi *Flutter* sebagai *framework* lintas platform yang hampir mendekati performa aplikasi *native*, serta mendukung berbagai arsitektur perangkat untuk *Android* dan *iOS* [34].

2.2.9 *Express Js*

ExpressJS adalah *framework NodeJS* yang dirancang untuk memudahkan dan mempercepat pembuatan aplikasi dan API, menggunakan pola desain fleksibel dan tanpa banyak dependensi tambahan [35]. *API (Application Programming Interface)* adalah sekumpulan prosedur yang memungkinkan aplikasi lain mengakses fungsi atau fitur suatu sistem secara terprogram. *API* dibangun untuk mempercepat proses pengembangan perangkat lunak dengan memungkinkan penggunaan kembali fitur-fitur yang sama tanpa perlu membuatnya dari awal. *API* menghubungkan basis data dengan

perangkat lunak, dan digunakan untuk mengeksekusi perintah pada sistem. *Web service*, yang berperan sebagai komponen penghubung antarmuka dan *server*, mendukung prosedur yang memproses request dan response melalui *API* [36].

2.2.10 *Mongo DB*

MongoDB adalah sistem manajemen basis data *NoSQL* yang populer, menggunakan model penyimpanan berbasis dokumen yang biasanya dalam format *JavaScript Object Notation (JSON)* atau *Binary JSON*. Hal ini memungkinkan penyimpanan dan akses data yang lebih fleksibel dan efisien, dibandingkan dengan model berbasis tabel yang digunakan oleh *Database* relasional. Keunggulan utama *MongoDB* terletak pada kemampuannya untuk menangani data dalam format *BSON*, yang mempercepat proses penyimpanan dan pengambilan data [37]. Karena kemampuannya menangani data yang sering berubah, *MongoDB* sangat cocok untuk aplikasi yang membutuhkan fleksibilitas dan skalabilitas tinggi, dan sering digunakan bersama dengan *Node.js* untuk membangun aplikasi *web* dinamis yang optimal [38].

2.2.11 *Firebase*

Firebase adalah platform pengembangan aplikasi *web* dan seluler yang menyediakan berbagai layanan *backend*, yang dirancang untuk membantu pengembang membangun aplikasi berkualitas. Platform ini memungkinkan pengembang membuat *API* yang dapat disinkronkan secara otomatis dengan berbagai klien berbeda, baik aplikasi berbasis *web* maupun *mobile*, serta menyimpan data di cloud *Firebase*. *Firebase* mendukung integrasi dengan berbagai bahasa dan

platform seperti *Android*, *iOS*, *JavaScript*, *Java*, *Objective-C*, dan *Node.js*, melalui pustaka yang mempermudah implementasi. Selain itu, *Firebase* memungkinkan akses *Database* melalui *REST API*, yang memanfaatkan protokol *Server-Sent Events (SSE)* untuk membuka koneksi *HTTP* agar *server* dapat mengirim notifikasi secara langsung ke klien. Pengembang dapat menggunakan *REST API* untuk mengirim data, sementara *library Firebase* yang diterapkan dalam aplikasi akan secara otomatis mengambil dan memperbarui data secara *real-time* [39]. Berikut ini adalah fitur yang tersedia di *Firebase*, antara lain [40]:

1. *Firebase Analytics*

FirebaseAnalytics dapat menyediakan analisis perilaku pengguna aplikasi untuk membantu pengembang memahami interaksi pengguna dengan aplikasi. Dengan fitur ini, pengembang dapat meningkatkan pengalaman pengguna dan meningkatkan efektivitas aplikasi.

2. *Firebase Cloud Messaging (FCM)*

Firebase Cloud Messaging memungkinkan pengembang untuk mengirim notifikasi atau pesan kepada pengguna aplikasi di platform *Android*, *iOS*, dan *web*. *FCM* mendukung pengiriman pesan notifikasi atau data dengan fleksibel, seperti mengirim pesan ke perangkat tunggal, grup, atau perangkat yang berlangganan topik.

3. *Firebase Auth*

Firebase Authentication menyediakan sistem autentikasi yang aman dan mudah diintegrasikan ke dalam aplikasi. Fitur ini memungkinkan pengembang untuk menangani pendaftaran, *login*, dan manajemen akun pengguna dengan berbagai metode, termasuk

autentikasi melalui *email/password*, nomor telepon, serta penyedia identitas gabungan yang populer seperti *Google*.

4. *Real-time Database*

Firebase Real-time Database merupakan penyimpanan data berbasis cloud yang memungkinkan sinkronisasi data secara langsung antara aplikasi dan *server*. Data disimpan dalam format *JSON* dan akan otomatis diperbarui di semua perangkat pengguna yang terhubung tanpa perlu sinkronisasi secara manual.

5. *Firebase Cloud Storage*

Firebase Cloud Storage adalah fitur yang memungkinkan pengembang untuk mengunggah, mengunduh, dan mengelola file seperti gambar, video, dan dokumen. Fitur ini menggunakan *Cloud Storage for Firebase*, sehingga memudahkan akses dan pengelolaan file melalui *Firebase* dan *Google Cloud*.

6. *Firebase Crash Reporting (Crashlytics)*

Firebase Crashlytics adalah fitur yang dapat membantu pengembang dalam mendeteksi, memprioritaskan, serta menyelesaikan masalah stabilitas yang dapat mengurangi kualitas aplikasi. *Crashlytics* akan mengelompokkan *error* yang berulang ke dalam daftar masalah yang mudah ditangani, menyediakan informasi kontekstual, serta menampilkan tingkat keparahan dan frekuensi *error*, sehingga memudahkan pengembang untuk mengidentifikasi masalah lebih cepat.

7. *Firebase Test Lab*

Firebase Test Lab adalah fitur pengujian aplikasi berbasis *cloud* yang memungkinkan pengembang untuk menguji aplikasi di

berbagai jenis perangkat dan konfigurasi, untuk memberikan gambaran yang lebih baik tentang performanya ketika digunakan oleh pengguna.

2.2.12 *Black Box Testing*

Pengujian *Black Box* adalah metode pengujian perangkat lunak yang menguji fungsionalitas suatu sistem tanpa memperhatikan detail internal kode atau logika pengoperasian perangkat lunak. Fokusnya adalah pada *input* dan *output* yang dihasilkan oleh perangkat lunak. Tujuan dari pendekatan ini adalah untuk memastikan bahwa perangkat lunak bekerja sesuai dengan harapan pengguna [12]. Dalam konteks pengujian aplikasi, metode pengujian *Black Box* digunakan untuk menemukan berbagai jenis kesalahan atau penyimpangan, termasuk kesalahan fungsional, kesalahan tampilan, kesalahan basis data eksternal, kesalahan kinerja, dan kesalahan penghentian. Metode ini membantu memastikan bahwa perangkat lunak dapat berfungsi dengan benar tanpa harus memahami detail implementasi internalnya [41].

2.2.13 *System Usability Scale*

Evaluasi produk perangkat lunak dapat dilakukan menggunakan *System Usability Scale (SUS)*, sebuah metode yang pertama kali diperkenalkan oleh John Brooke pada tahun 1986. *SUS* digunakan untuk menilai tingkat kegunaan (*usability*) dari beragam produk atau layanan, termasuk perangkat keras, perangkat lunak, aplikasi *mobile*, situs *web*, dan lainnya [13]. Dalam praktiknya, jumlah minimal responden yang digunakan adalah 30 orang. Hal ini sejalan dengan pendapat Sugiyono [42], yang menyatakan bahwa uji validitas

dan reliabilitas dapat dilakukan dengan minimal 30 responden guna memperoleh hasil yang mendekati distribusi normal.

SUS terdiri dari sepuluh pernyataan yang terdiri dari lima pernyataan positif dan lima pernyataan negatif yang mencakup aspek-aspek penting dalam *usability*, seperti kemudahan penggunaan, kejelasan, dan konsistensi antarmuka [13]. Berikut ini adalah sepuluh pernyataan yang terdapat pada *SUS*, yaitu:

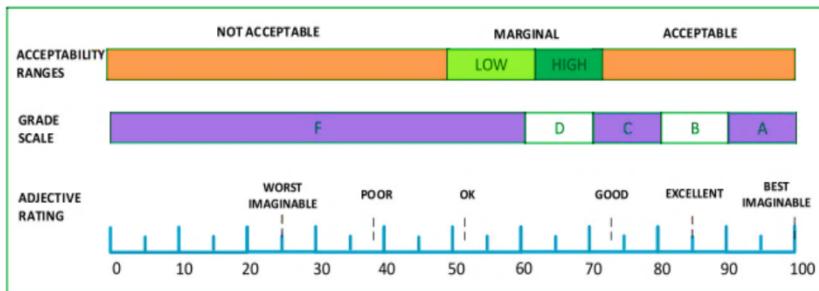
1. Saya berfikir akan menggunakan aplikasi ini lagi.
2. Saya merasa aplikasi ini rumit untuk digunakan.
3. Saya merasa aplikasi ini mudah untuk digunakan.
4. Saya membutuhkan bantuan dari orang lain atau teknisi dalam menggunakan aplikasi ini.
5. Saya merasa fitur-fitur aplikasi ini berjalan dengan semestinya.
6. Saya merasa ada banyak hal yang tidak konsisten (tidak serasi) pada aplikasi ini.
7. Saya merasa orang lain akan memahami cara menggunakan aplikasi ini dengan cepat.
8. Saya merasa aplikasi ini membingungkan.
9. Saya merasa tidak ada hambatan dalam menggunakan aplikasi ini.
10. Saya perlu membiasakan terlebih dahulu sebelum menggunakan aplikasi ini.

Untuk menggunakan *System Usability Scale (SUS)*, diperlukan skala *Likert* untuk menilai tingkat persetujuan atau ketidaksetujuan responden terhadap pernyataan yang diberikan. Skala *Likert* ini memberikan pilihan nilai dengan rentang 1 (sangat tidak setuju) hingga 5 (sangat setuju) untuk masing-masing pernyataan. Untuk pernyataan *SUS* yang bernomor 1, 3, 5, 7, dan 9 atau ganjil, akan dihitung dengan

cara mengurangi nilai Likert dengan 1. Sedangkan untuk pernyataan *SUS* yang bernomor 2, 4, 6, 8, dan 10 atau genap, akan dihitung dengan cara mengurangkan nilai Likert dari 5. Total skor dihitung dengan menjumlahkan semua nilai dari sepuluh pernyataan. kemudian dikalikan dengan 2,5 untuk memperoleh nilai akhir dari penilaian *SUS* [43]. Rumus keseluruhan perhitungan nilai *SUS* adalah:

$$\text{Skor Akhir} = (\sum (\text{Skor Ganjil} + \text{Skor Genap})) \times 2.5 \quad (2.2)$$

Skor akhir dari *System Usability Scale (SUS)* memberikan gambaran tentang tingkat usabilitas produk, dengan rentang nilai antara 0 hingga 100. Skor yang diperoleh dari perhitungan ini dapat diinterpretasikan ke dalam kategori-kategori tertentu, yang dapat dilihat pada gambar 2.2 berikut:



Gambar 2.2 Kategori dan Rentang Nilai Skor *SUS* [44]

Tabel 2.7 dibawah ini dapat memberikan gambaran yang lebih jelas dan mudah dipahami mengenai nilai-nilai interpretasi hasil *SUS* [45].

Tabel 2.6 Skala Interpretasi Hasil *SUS*

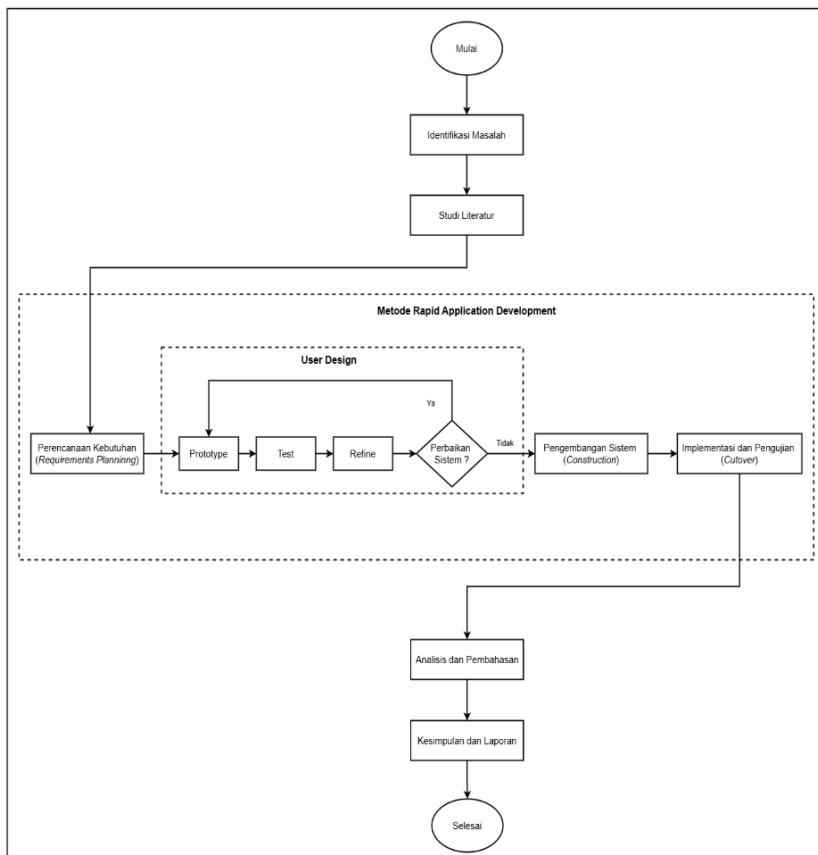
Skor <i>SUS</i>	Grade Scale	Adjective Rating	Acceptability Ranges	
84,1 – 100	A+	<i>Best imaginable</i>	<i>Acceptable</i>	
80,8 – 84,0	A	<i>Excellent</i>		
78,9 – 80,9	A-	<i>Good</i>		
77,2 – 78,8	B+			
74,1 – 77,1	B			
72,6 – 74,0	B-			
71,1 – 72,5	C+			
65,0 – 71,0	C	<i>OK</i>	<i>Marginal</i>	
62,7 – 64,9	C-			
51,7 – 62,6	D			
25,1 – 51,6	F	<i>Poor</i>	<i>Not Acceptable</i>	
0 – 25,0	F-	<i>Worst imaginable</i>		

BAB III

METODE PENELITIAN

3.1 Alur Penelitian

Alur penelitian dalam mengembangkan aplikasi monitoring dan kontrol otomatis berbasis *Internet of Things (IoT)* untuk budidaya udang dengan menggunakan metode *Rapid Application Development (RAD)* digambarkan dalam sebuah *diagram flowchart* yang dapat dilihat pada gambar 3.1 berikut.



Gambar 3.1 Alur Metode Penelitian

3.2 Penjabaran Langkah Penelitian

Dari alur penelitian yang telah dibuat sebelumnya berikut ini penjelasan detail dari tiap tahapan alur penelitian.

3.2.1 Identifikasi Masalah

Seperti penjelasan pada latar belakang, masih banyak pembudidaya udang yang menggunakan metode pengelolaan tambak secara manual, termasuk dalam pemantauan kualitas air dan pemberian pakan. Hal ini sering mengakibatkan berbagai kendala yang dapat menurunkan produktivitas, seperti ketidakakuratan dalam mengukur kualitas air, serta pemberian pakan yang tidak merata. Kualitas air yang tidak terpantau dengan baik dapat berdampak pada pertumbuhan dan kesehatan benih udang, sedangkan pemberian pakan yang tidak teratur dapat menyebabkan pertumbuhan udang yang tidak optimal. Oleh karena itu, diperlukan system aplikasi monitoring dan kontrol otomatis yang dapat meningkatkan efisiensi dan efektivitas dalam budidaya udang.

3.2.2 Studi Literatur

Studi literatur dilaksanakan dengan cara meneliti buku, mencari jurnal nasional atau internasional, serta menelusuri tugas akhir terdahulu melalui *internet*. Literatur yang diakses bertujuan untuk mendukung penelitian terkait pengembangan aplikasi monitoring dan control otomatis budidaya udang dan mencari referensi terkait metode yang digunakan, khususnya *RAD*. Tahap ini dilakukan agar dalam pengembangan aplikasi akan lebih terarah serta efektif.

3.2.3 Penerapan Metode *Rapid Application Development (RAD)*

Berdasarkan permasalahan yang ada, pengembangan aplikasi ini akan menggunakan metode *Rapid Application Development (RAD)*. Metode *RAD* dipilih karena fokus pada pengembangan aplikasi yang cepat dengan melibatkan umpan balik pengguna secara intensif dalam setiap fase pengembangannya. Dalam penerapan *RAD*, terdapat empat tahapan meliputi Perencanaan Kebutuhan (*Requirements Planning*), Desain Pengguna (*User Design*), Pengembangan Sistem (*Construction*), Implementasi (*Cutover*). Dengan cara ini, pengembangan aplikasi menjadi lebih fleksibel dan cepat karena dapat segera menyesuaikan berdasarkan umpan balik. Dengan penerapan *RAD*, diharapkan aplikasi yang dikembangkan dapat memenuhi kebutuhan pengguna dengan lebih akurat dan efektif dalam waktu yang relatif singkat.

3.2.4 Analisis dan Pembahasan

Pada tahap ini, peneliti melakukan analisis terhadap data dan hasil yang diperoleh selama proses penelitian. Analisis ini bertujuan untuk mengevaluasi sejauh mana penerapan metode *Rapid Application Development (RAD)* dalam pengembangan aplikasi monitoring dan kontrol otomatis tambak udang dapat memenuhi kebutuhan yang telah diidentifikasi. Peneliti akan menjelaskan hasil implementasi dari metode yang digunakan. Selain itu, tahap ini juga mencakup pembahasan mengenai performa aplikasi yang dikembangkan, respons pengguna terhadap fitur yang telah diterapkan, serta potensi pengembangan lebih lanjut untuk meningkatkan kualitas dan fungsionalitas aplikasi.

3.2.5 Kesimpulan dan Laporan

Pada tahap ini, peneliti menyusun kesimpulan berdasarkan hasil analisis dan pembahasan yang telah dilakukan. Kesimpulan ini memberikan gambaran singkat mengenai efektivitas metode *RAD* dalam pengembangan aplikasi monitoring dan kontrol otomatis tambak udang serta sejauh mana aplikasi yang dihasilkan dapat menjawab kebutuhan yang telah diidentifikasi pada tahap awal penelitian. Selain itu, laporan penelitian juga disusun sebagai dokumentasi lengkap dari seluruh proses yang telah dilakukan, mulai dari perencanaan hingga implementasi dan evaluasi. Laporan ini mencakup metodologi yang digunakan, hasil pengujian, serta rekomendasi untuk pengembangan lebih lanjut, sehingga dapat menjadi referensi bagi penelitian atau proyek serupa di masa mendatang.

3.3 Alat dan Bahan Tugas akhir

Dalam penelitian ini, diperlukan berbagai alat untuk menyelesaikan pengembangan aplikasi. Alat yang digunakan untuk mendukung pengembangan dan implementasi aplikasi akan dijelaskan pada sub-bab berikut ini.

3.3.1 Alat

Alat yang digunakan pada pengembangan aplikasi monitoring dan control otomatis budidaya udang adalah sebagai berikut:

1. *Laptop* dengan spesifikasi sistem operasi *Windows 10*, processor *Intel Core i3 10110U CPU @ 2,10 GHz*, memori 4 GB, SSD 258 GB
2. *Smartphone* dengan spesifikasi *OS Android minimal 11*, dengan penyimpanan Internal 32 GB, dan 3 GB RAM.

3. *Figma*, digunakan untuk desain tampilan Aplikasi.
4. *Android Studio*, digunakan untuk memudahkan pengembangan Aplikasi.
5. *Github*, digunakan untuk menyimpan file – file atau data – data yang digunakan dalam pengembangan sistem.
6. Bahasa Pemrograman *Flutter (Dart)*, digunakan membangun Aplikasi.
7. *Firebase* Sebagai media koneksi yang menyambungkan aplikasi *mobile* , *website* dan *IoT*.
8. *Express Js*, digunakan untuk membangun *API* dan layanan *backend* yang menghubungkan aplikasi *mobile* dengan *MongoDB* dan perangkat *IoT*, memungkinkan pengelolaan data secara lebih fleksibel.
9. *MongoDB*, digunakan untuk menyimpan data aplikasi.

3.3.2 Bahan

Bahan yang digunakan pada pengembangan aplikasi monitoring dan control otomatis budidaya udang adalah sebagai berikut:

1. Hasil wawancara kebutuhan aplikasi *mobile* yang dilakukan dengan pemilik tambak udang yang tergabung di kelompok budidaya udang Sadewa Farm.

3.4 Metode Pengembangan

Pada penelitian ini, metode pengembangan yang digunakan ialah *Rapid Application Development (RAD)*. Metode ini berfokus pada pengembangan aplikasi yang cepat dengan melibatkan umpan balik pengguna secara intensif dalam setiap fase pengembangannya melalui

beberapa tahapan utama berupa Perencanaan Kebutuhan, Desain Pengguna, Pengembangan Sistem, dan Implementasi. Adapun penjabaran tahapan-tahapan dari metode *Rapid Application Development* adalah sebagai berikut.

3.4.1 Perencanaan Kebutuhan (*Requirements Planning*)

Tahap ini adalah langkah awal dalam metode pengembangan sistem *Rapid Application Development (RAD)*. Pada tahap ini, kebutuhan dari calon pengguna dikumpulkan melalui observasi dan wawancara langsung dengan pembudidaya udang. Proses ini bertujuan untuk memahami secara mendalam kebutuhan dan harapan pengguna terhadap sistem aplikasi yang akan dikembangkan, sehingga aplikasi yang dibangun dapat memenuhi ekspektasi mereka dengan efektif. Berikut pertanyaan dan jawaban wawancara yang dijadikan peneliti sebagai dasar untuk mendefinisikan kebutuhan sistem pada penelitian ini dapat dilihat melalui tabel 3.1 berikut.

Tabel 3.1 Hasil Wawancara

No	Pertanyaan	Jawaban
1.	Apa tantangan terbesar yang dihadapi dalam mengelola tambak udang?	Terkadang kami kesulitan dalam menjaga kualitas air di tambak, terutama saat musim hujan. Kami kesulitan dalam memantau kondisi kualitas air pada tambak dengan tepat karena kami hanya bergantung pada pemeriksaan sederhana.

No	Pertanyaan	Jawaban
2.	Apakah pengelolaan tambak seperti memantau kualitas air atau pemberian pakan saat ini menggunakan cara manual atau otomatis?	Saat ini, semuanya masih dilakukan secara manual. Kami memeriksa kualitas air dan memberikan pakan secara langsung, tanpa bantuan alat. Jadi memerlukan banyak tenaga dan waktu.
3.	Parameter kualitas air apa saja yang ingin dipantau dalam tambak udang?	Kami perlu memantau suhu air, pH, salinitas, dan kadang juga kekeruhan air.
4.	Apakah Bapak tertarik untuk mengotomatisasi beberapa fungsi di tambak, seperti <i>aerator</i> dan pemberian pakan?	Iya, saya tertarik sekali. Jika bisa mengotomatisasi <i>aerator</i> dan pemberian pakan, itu akan sangat membantu. Jadi bisa fokus pada pekerjaan lain yang lebih penting.
5.	Jika ada aplikasi yang dapat mengontrol perangkat tambak secara langsung, apakah bapak tertarik?	Tentu, dengan aplikasi seperti itu, saya bisa mengontrol perangkat tambak dari jarak jauh, sehingga lebih efisien dan tidak perlu selalu ada di tambak.
6.	Fitur apa saja yang paling dibutuhkan dalam aplikasi	Fitur yang paling dibutuhkan adalah pemantauan kualitas air secara <i>real-time</i> ,

No	Pertanyaan	Jawaban
	untuk memudahkan pengelolaan tambak udang?	pemberitahuan otomatis jika ada masalah, serta kontrol perangkat seperti <i>aerator</i> dan pembarian pakan lewat aplikasi. Fitur laporan berkala data kualitas air tambak juga akan sangat membantu.

Setelah hasil observasi dan wawancara diperoleh, peneliti dapat merumuskan kebutuhan pengguna secara lebih terstruktur. Dari proses ini akan diperoleh kebutuhan fungsional dan kebutuhan non-fungsional.

3.2.3.1 Kebutuhan Fungsional

Kebutuhan fungsional mendeskripsikan fitur-fitur utama yang harus ada di dalam aplikasi. Berikut merupakan daftar kebutuhan fungsional yang akan dikembangkan dan diimplementasikan dapat dilihat pada tabel 3.2.

Tabel 3.2 Daftar Kebutuhan Fungsional

ID	Kebutuhan Fungsional
F-001	Aplikasi dapat melakukan <i>login</i> dan <i>logout</i> .
F-002	Admin dapat mengelola pengguna aplikasi, seperti menambahkan <i>user</i> baru, edit <i>user</i> dan menghapus <i>user</i> yang sudah ada.
F-003	Pengguna dapat melakukan <i>reset password</i> jika pengguna lupa kata sandi mereka.

ID	Kebutuhan Fungsional
F-004	Pengguna dapat menambahkan, menedit dan menghapus kolam yang akan dipantau dan dikontrol melalui aplikasi.
F-006	Pengguna dapat mengedit data pribadi di <i>profile</i> .
F-007	Aplikasi dapat menyajikan parameter kualitas air seperti suhu, pH, salinitas, dan tingkat kekeruhan secara <i>real-time</i> .
F-008	Aplikasi dapat menampilkan grafik interaktif untuk setiap parameter kualitas air.
F-009	Aplikasi dapat mengatur batasan parameter masing-masing sensor (suhu, pH, salinitas, dan tingkat kekeruhan).
F-010	Aplikasi dapat mengirimkan notifikasi jika parameter lingkungan tambak (suhu, pH, salinitas, kekeruhan) melewati batas aman yang telah ditetapkan.
F-011	Aplikasi dapat mengirimkan notifikasi jika ada perubahan data pada batasan parameter sensor, jadwal pemberian pakan, jumlah pakan dan <i>Delay aerator</i> .
F-012	Aplikasi dapat mengirimkan notifikasi jika jumlah pakan dalam tabung hampir habis.
F-013	Aplikasi dapat menghasilkan laporan yang merangkum data riwayat parameter kualitas air

ID	Kebutuhan Fungsional
	untuk jangka waktu harian yang bisa dilihat dan diunduh oleh pengguna.
F-014	Aplikasi dapat memberikan informasi seputar perangkat sensor.
F-015	Aplikasi dapat mengontrol secara otomatis pelontar pakan dan <i>aerator</i> berdasarkan waktu yang telah ditentukan.
F-016	Pengguna dapat mengontrol pelontar pakan dan <i>aerator</i> sesuai dengan kebutuhan melalui aplikasi.
F-017	Pengguna dapat mengatur waktu pemberian pakan dan <i>aerator</i> melalui aplikasi.
F-018	Pengguna dapat mengatur jumlah pakan udang melalui aplikasi.

3.2.3.2 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional merupakan batasan fungsi pada aplikasi yang telah dikembangkan. Berikut merupakan daftar kebutuhan non-fungsional yang telah didefinisikan dapat dilihat pada tabel 3.3.

Tabel 3.3 Daftar Kebutuhan Non-Fungsional

ID	Parameter	Kebutuhan Non-Fungsional
NF-001	<i>Portabilitas</i>	Aplikasi dapat dijalankan di perangkat <i>mobile</i> dengan sistem operasi <i>Android</i> .

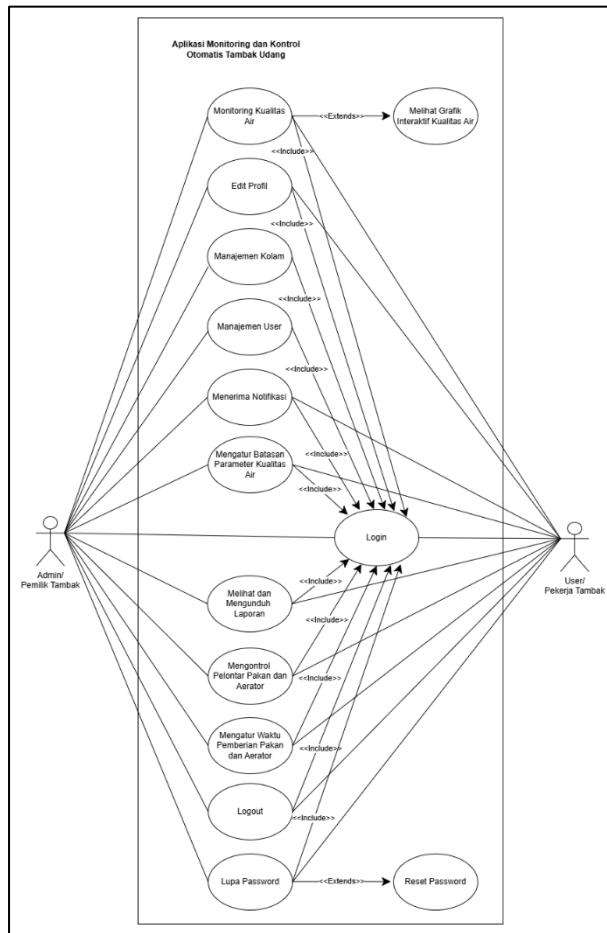
ID	Parameter	Kebutuhan Non-Fungsional
NF-002	<i>Kompatibilitas</i>	Aplikasi dapat menampilkan data dari perangkat IoT.
NF-003	<i>Security</i>	Akses pengguna dibatasi sesuai peran (<i>role-based access control</i>).

3.4.2 Desain Pengguna (*User Design*)

Tahap Desain Pengguna melibatkan kolaborasi antara pengguna dan pengembang untuk merancang antarmuka dan interaksi sistem berdasarkan kebutuhan yang telah diidentifikasi pada tahap sebelumnya. Pada tahap ini, akan dibuat berbagai model dan *diagram* seperti *Use case Diagram*, *Activity Diagram*, *Class Diagram*, *Object Diagram* serta merancang tampilan antarmuka pengguna.

3.4.2.1 *Use case Diagram*

Diagram ini dimanfaatkan untuk mengilustrasikan hubungan antara sistem dan aktor (pengguna). *Use case diagram* memberikan pemahaman yang terperinci tentang fungsi sistem dari sudut pandang pengguna dan sistem. Berikut adalah *use case diagram* untuk aplikasi monitoring dan kontrol otomatis budidaya udang.



Gambar 3.2 Use case Diagram

Pada gambar 3.2 di perlihatkan *use case* pada aplikasi monitoring dan kontrol otmatis pada budidaya udang dirancang untuk memantau kualitas air dan pengelolaan pakan secara optimal. Dalam aplikasi ini, terdapat dua aktor utama, yaitu Admin sebagai pemilik tambak dan *User* sebagai pekerja tambak. Admin memiliki akses untuk *login*, *logout*, *reset password*, *edit profile*, manajemen kolam, manajemen *user*, memantau parameter kualitas air, mengelola

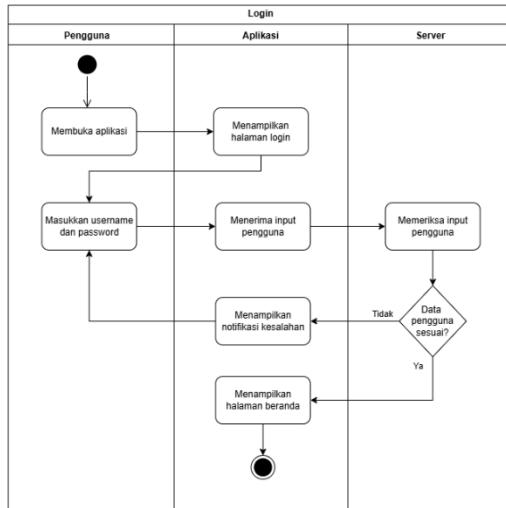
pengguna, menerima notifikasi, mengubah batasan parameter kualitas air, melihat dan mengunduh laporan riwayat kualitas air, serta mengendalikan dan mengatur waktu pemberian pakan dan *aerator*. Sedangkan, *user* juga memiliki akses yang sama dengan admin seperti untuk *login*, *logout*, *reset password*, *edit profile*, memantau parameter kualitas air, menerima notifikasi terkait parameter air, mengubah batasan parameter kualitas air, melihat dan mengunduh laporan riwayat kualitas air, serta mengendalikan dan mengatur waktu pemberian pakan dan *aerator*, namun *user* tidak dapat melakukan manajemen kolam dan manajemen *user*.

Semua aktivitas memerlukan *login* terlebih dahulu, yang diindikasikan oleh hubungan <>include<> antara *use case* ‘*Login*’ dengan *use case* lainnya. *Use case* ‘Monitoring parameter air’ memiliki hubungan <>extend<> dengan ‘Melihat grafik historis parameter air’. yang berarti fitur ini merupakan ekstensi dari monitoring parameter air yang memungkinkan pengguna untuk melihat grafik data historis dari parameter kualitas air yang dimonitor. Selain itu, *use case* ‘*Lupa Password*’ juga memiliki hubungan <>extend<> dengan ‘*Reset password*’, yang menunjukkan bahwa *reset password* hanya terjadi ketika pengguna mengalami kesulitan mengingat *password* mereka, dan ini merupakan langkah tambahan dalam alur *login*.

3.4.2.2 *Activity Diagram*

Activity Diagram menggambarkan alur kerja atau kegiatan dalam sistem atau proses bisnis. Ilustrasi di bawah ini menampilkan aktivitas yang dilakukan oleh pengguna dan admin dalam sistem aplikasi monitoring dan kontrol otomatis budidaya udang.

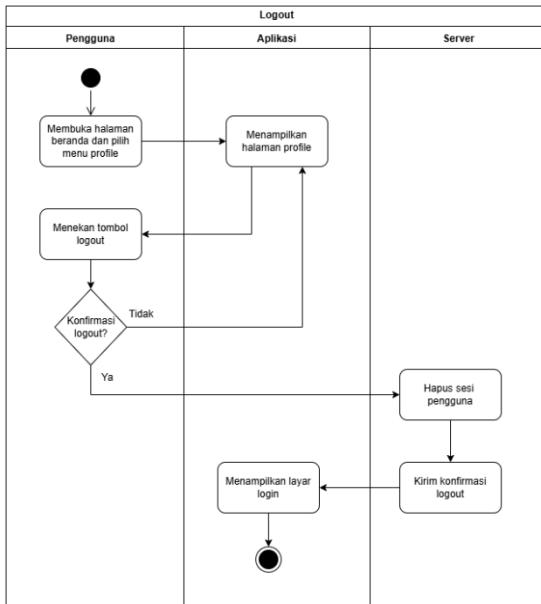
1. Activity Diagram – Login



Gambar 3.3 Activity Diagram Login

Pada gambar 3.3 ini diperlihatkan, alur proses *Activity Diagram login* dimulai dari pengguna yang akan melakukan *login* harus memasukkan *username* dan *password*. Setelah itu, aplikasi menerima *inputan* dari pengguna dan meneruskan data tersebut ke *server* untuk dilakukan proses validasi. *Server* kemudian memeriksa data yang dimasukkan dan mencocokkannya dengan data yang tersimpan di dalam basis data. Selanjutnya, *server* memutuskan apakah data yang dimasukkan oleh pengguna sesuai atau tidak. Jika data tidak sesuai, aplikasi akan menampilkan notifikasi kesalahan kepada pengguna, yang menunjukkan bahwa *username* atau *password* yang dimasukkan salah. Sebaliknya, jika data pengguna sesuai, aplikasi akan menampilkan halaman beranda sebagai indikasi bahwa pengguna berhasil *login* ke dalam aplikasi.

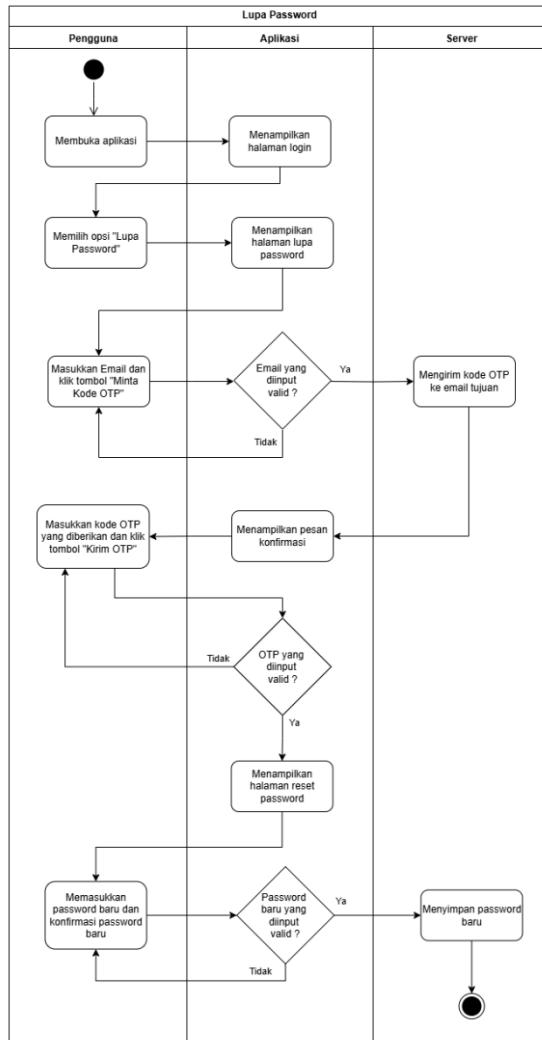
2. Activity Diagram – Logout



Gambar 3.4 Activity Diagram Logout

Pada gambar 3.4 ini diperlihatkan, alur proses *Activity Diagram logout* dimulai dari pengguna yang akan melakukan *logout* harus membuka halaman beranda dan pilih menu *profile* untuk masuk ke halaman *profile*. Kemudian, pengguna menekan tombol *logout* dan pengguna akan mengkonfirmasi bahwa akan *logout* dan konfirmasi tersebut ke *server* untuk dilakukan penghapusan sesi pengguna. *Server* kemudian mengirim konfirmasi bahwa pengguna akan *logout* ke aplikasi, yang kemudian aplikasi akan menerima konfirmasi tersebut dan menampilkan layar *login*. Jika pengguna tidak mengkonfirmasi untuk *logout*, maka pengguna akan diarahkan ke halaman *profile*.

3. Activity Diagram – Lupa Password

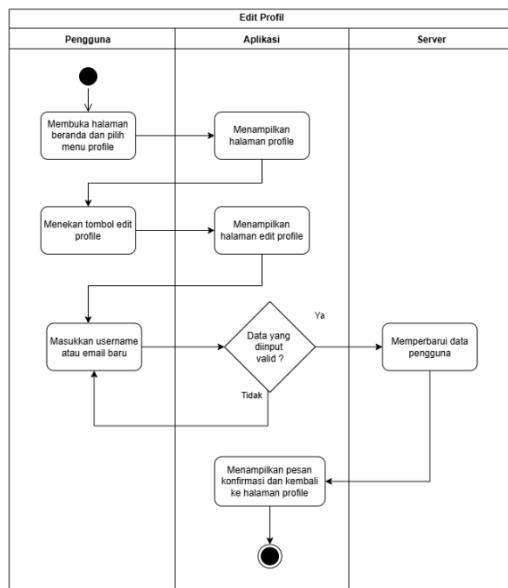


Gambar 3.5 Activity Diagram Lupa Password

Pada gambar 3.5 ini diperlihatkan, alur proses *Activity Diagram* lupa password dimulai dari pengguna yang lupa password saat akan melakukan *login* dapat memilih opsi “Lupa Password” untuk *reset password*. Pengguna akan diarahkan ke

halaman lupa *password* dan harus memasukkan *email* untuk meminta kode *OTP*. Kemudian *email* akan divalidasi oleh aplikasi, setelah divalidasi aplikasi akan menampilkan pesan konfirmasi bahwa kode *OTP* telah terkerim ke *email*. Pengguna memasukkan kode *OTP* yang diberikan dan menekan tombol kirim *OTP*, aplikasi akan memvalidasi *OTP* yang diinput, jika valid, pengguna akan masuk ke halaman ganti *password*. Pengguna akan diminta memasukkan *password* baru dan mengkonfirmasi *password* baru tersebut. Kemudian, aplikasi akan memvalidasi *password* yang baru diberikan, sebelum *password* baru disimpan ke dalam *Database*.

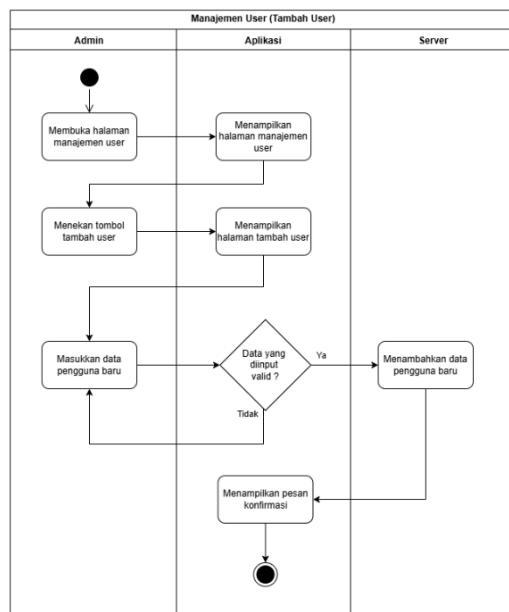
4. Activity Diagram – Edit Profile



Gambar 3.6 Activity Diagram Edit Profile

Pada gambar 3.6 ini diperlihatkan, alur proses *Activity Diagram* edit *profile* dimulai dari pengguna membuka halaman beranda dan pilih menu *profile* untuk masuk ke halaman *profile*. Pengguna diharuskan menekan tombol edit *profile* untuk mengedit *profile* pengguna. Aplikasi akan menampilkan halaman edit *profile* dan pengguna dapat memasukkan *username* atau *email* yang baru. Aplikasi akan memvalidasi data yang diinput, server akan memperbarui data pengguna di *Database* jika data yang diinput valid. Kemudian, aplikasi akan menampilkan pesan konfirmasi bahwa pengguna telah mengedit data *profile* dan aplikasi akan kembali ke halaman *profile*.

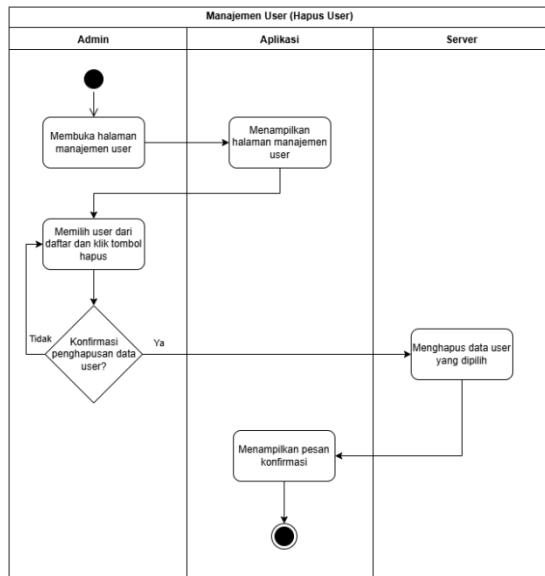
5. *Activity Diagram – Manajemen User (Tambah User)*



Gambar 3.7 *Activity Diagram Manajemen User (Tambah User)*

Pada gambar 3.7 ini diperlihatkan, alur proses *Activity Diagram* tambah *user* pada manajemen *user* dimulai dari admin membuka halaman manajemen *user*. Pada halaman manajemen *user*, admin diharuskan menekan tombol tambah *user* untuk menambahkan *user* baru yang dapat mengkases aplikasi. Aplikasi akan menampilkan halaman tambah *user* dan admin dapat memasukkan data pengguna baru. Data yang diinputkan akan di validasi oleh aplikasi, jika data tidak valid aplikasi akan meminta admin untuk memasukkan data pengguna baru yang valid, jika data valid, *server* akan menambahkan data pengguna baru ke *Database*. Kemudian, aplikasi akan menampilkan pesan konfirmasi bahwa data *user* baru telah ditambahkan.

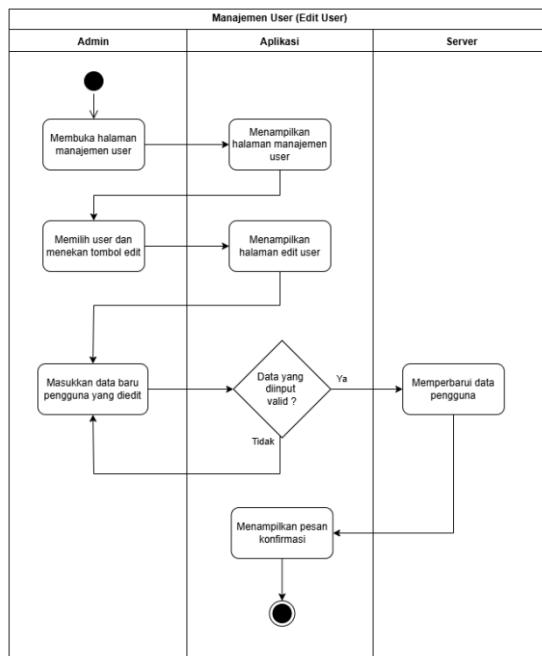
6. *Activity Diagram – Manajemen User (Hapus User)*



Gambar 3.8 *Activity Diagram* Manajemen User (Hapus User)

Pada gambar 3.8 ini diperlihatkan, alur proses *Activity Diagram* hapus *user* pada manajemen *user* dimulai dari admin membuka halaman manajemen *user*. Pada halaman manajemen *user*, admin diharuskan memilih *user* yang akan dihapus dan kemudian menekan tombol hapus. *Server* akan menghapus data yang telah dipilih dan dikonfirmasi oleh admin. Kemudian, aplikasi akan menampilkan pesan konfirmasi bahwa data *user* telah dihapus.

7. *Activity Diagram – Manajemen User (Edit User)*

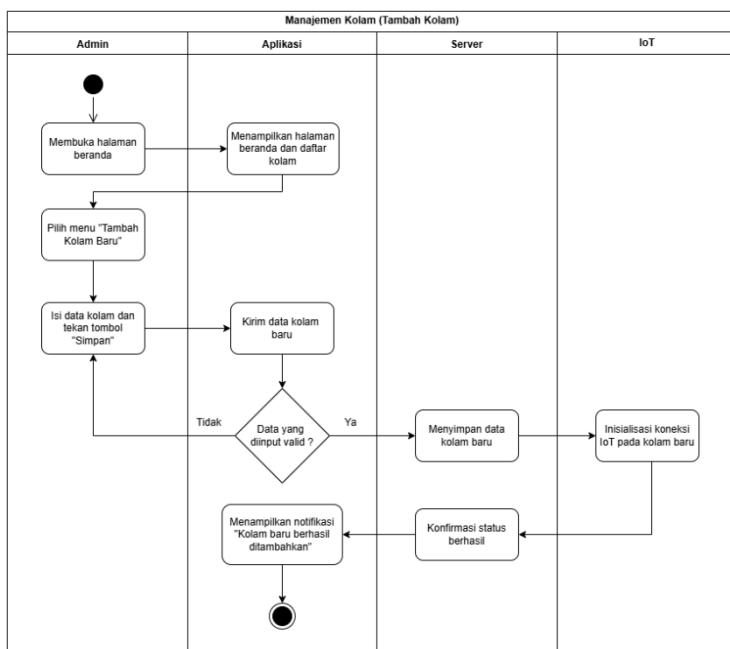


Gambar 3.9 *Activity Diagram* Manajemen *User* (Edit *User*)

Pada gambar 3.9 ini diperlihatkan, alur proses *Activity Diagram* edit *user* pada manajemen *user* dimulai dari admin membuka halaman manajemen *user*. Pada halaman manajemen

user, admin diharuskan memilih *user* yang akan diedit dan kemudian menekan tombol edit. Admin akan memasukkan data pengguna yang akan diperbarui, kemudian aplikasi akan mengkonfirmasi data yang *diinput*. Kemudian, aplikasi akan mengirim pembaruan data ke *server* untuk memperbarui data pengguna dan aplikasi akan menampilkan pesan bahwa data pengguna berhasil diperbarui.

8. Activity Diagram – Manajemen Kolam (Tambah Kolam)

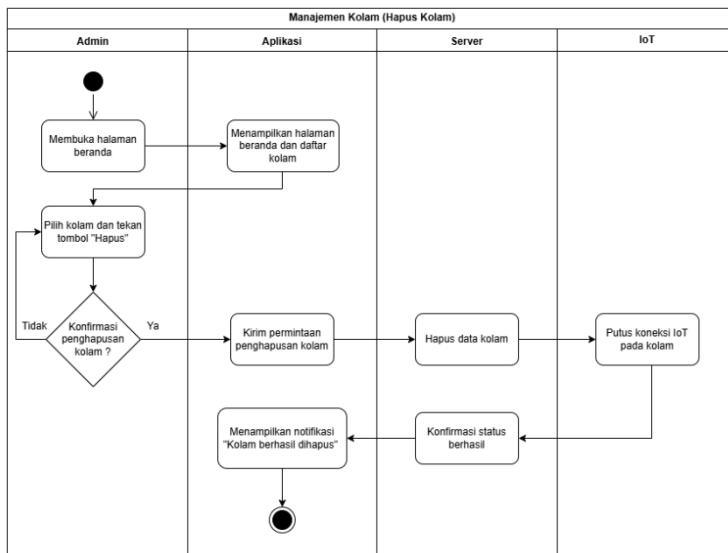


Gambar 3.10 *Activity Diagram* Manajemen Kolam (Tambah Kolam)

Pada gambar 3.10 ini diperlihatkan, alur proses *Activity Diagram* tambah kolam pada manajemen kolam dimulai dari admin membuka halaman beranda. Kemudian, aplikasi akan

menampilkan halaman beranda yang berisi daftar kolam, admin diharuskan memilih menu tambah kolam baru dan melengkapi data kolam baru, lalu menekan tombol simpan. Aplikasi akan mengirim data yang *diinput* admin ke *server*, namun sebelum sampai ke *server*, aplikasi akan memvalidasi data yang *diinput*. Aplikasi akan melanjutkan pengiriman data kolam baru ke *server* dan menyimpan data kolam yang baru. Selanjutnya, *IoT* akan menginisialisasi koneksi perangkat monitoring dan kontrol otomatis pada kolam baru. *Server* akan mengkonfirmasi status koneksi, jika berhasil aplikasi akan menampilkan notifikasi bahwa kolam baru berhasil ditambahkan.

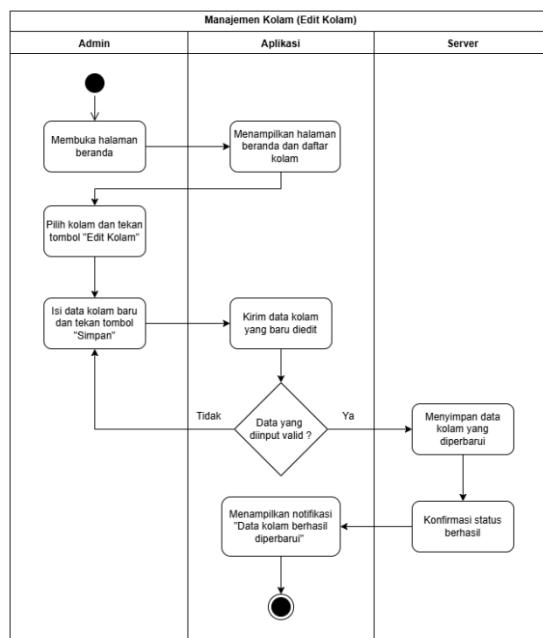
9. Activity Diagram – Manajemen Kolam (Hapus Kolam)



Gambar 3.11 *Activity Diagram* Manajemen Kolam (Hapus Kolam)

Pada gambar 3.11 ini diperlihatkan, alur proses *Activity Diagram* hapus kolam pada manajemen kolam dimulai dari admin membuka halaman beranda. Kemudian, aplikasi akan menampilkan halaman beranda yang berisi daftar kolam, admin diharuskan memilih kolam yang akan dihapus dan menekan tombol hapus. Admin akan mengkonfirmasi kolam yang dipilih. Selanjutnya *server* akan menghapus kolam yang dipilih admin dan *IoT* akan memutus koneksi pada perangkat monitoring dan kontrol otomatis pada kolam yang dihapus. Kemudian, *server* akan mengkonfirmasi status pemutusan koneksi, jika berhasil aplikasi akan menampilkan notifikasi bahwa kolam berhasil dihapus.

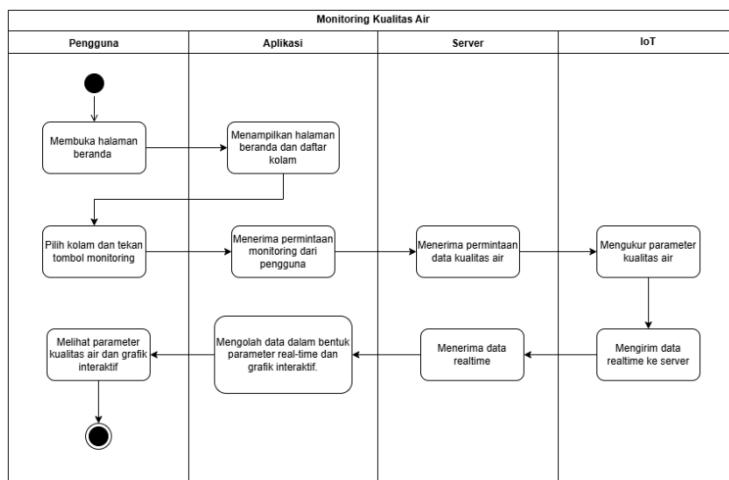
10. *Activity Diagram* – Manajemen Kolam (Edit Kolam)



Gambar 3.12 *Activity Diagram* Manajemen Kolam (Edit Kolam)

Pada gambar 3.12 ini diperlihatkan, alur proses *Activity Diagram* edit kolam pada manajemen kolam dimulai dari admin membuka aplikasi. Kemudian, aplikasi akan menampilkan daftar kolam, admin diharuskan memilih kolam yang akan di edit dan setelah memilih admin menekan tombol edit. Admin mengisi data baru pada kolam yang di edit, kemudian menekan tombol simpan. Aplikasi akan mengirim data kolam yang baru di edit dan melakukan validasi. *Server* akan mengkonfirmasi bahwa data berhasil diperbarui, kemudian aplikasi akan menampilkan notifikasi bahwa data kolam berhasil diperbarui

11. *Activity Diagram* – Monitoring Kualitas Air

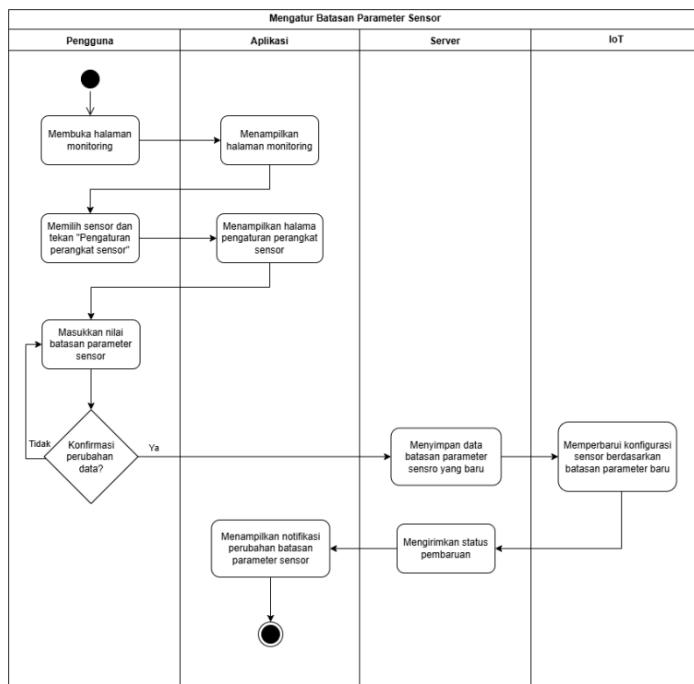


Gambar 3.13 *Activity Diagram* Monitoring Kualitas Air

Pada gambar 3.13 ini diperlihatkan, alur proses *Activity Diagram* monitoring kualitas air dimulai dari pengguna membuka halaman beranda. Aplikasi akan menampilkan halaman beranda dan daftar kolam yang dapat dipilih pengguna. Pengguna menekan

tombol monitoring pada kolam yang dipilih. Aplikasi akan mengirim permintaan tersebut ke *server*. *Server* akan melanjutkan permintaan aplikasi ke perangkat *IoT*, perangkat *IoT* akan mengukur kualitas air pada tambak udang dan mengirimkan nilai parameter sensor tersebut ke *server* secara relitime. *Server* akan menerima data dari perangkat *IoT* dan mengirimkannya ke aplikasi. Aplikasi akan mengolah data kualitas air dalam bentuk garfik yang interatif. Kemudian pengguna dapat melihat kualitas air yang telah diolah oleh aplikasi.

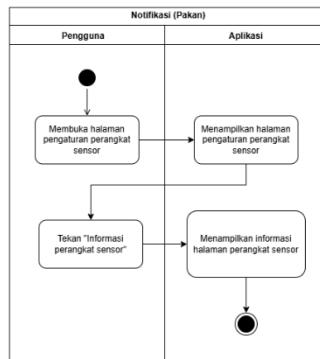
12. Activity Diagram – Mengatur Batasan Parameter Sensor



Gambar 3.14 *Activity Diagram* Mengatur Batasan Parameter Sensor

Pada gambar 3.14 ini diperlihatkan, alur proses *Activity Diagram* mengatur batasan parameter sensor dimulai dari pengguna membuka halaman monitoring. Aplikasi akan menampilkan halaman monitoring dan pengguna dapat menekan tombol pengaturan perangkat sensor. Aplikasi akan menampilkan halaman pengaturan perangkat sensor. Pengguna dapat mengubah nilai batas maksimal dan minimal pada sensor yang dipilih. Pengguna akan mengkonfirmasi perubahan data dan *server* akan menyimpan data batasan parameter sensor yang telah dikonfirmasi ke *Database*. Kemudian, *server* akan mengirim batasan yang baru ke perangkat *IoT* untuk di konfigurasikan. Setelah konfigurasi, *server* akan mengirim status pembaruan ke aplikasi. Sehingga aplikasi akan menampilkan notifikasi bahwa batasan parameter sensor telah diubah.

13. *Activity Diagram* – Melihat Informasi Perangkat Sensor

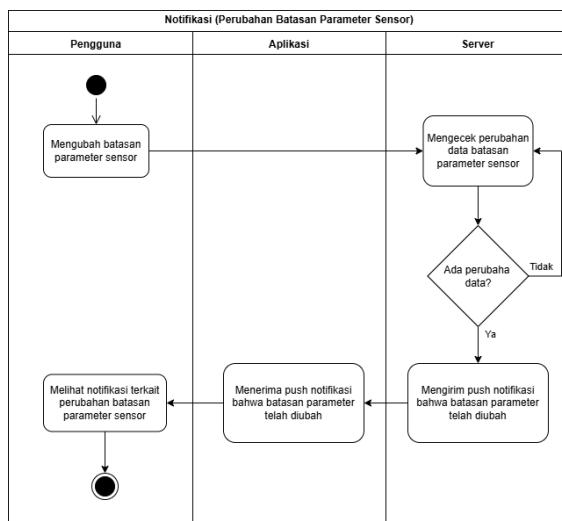


Gambar 3.15 *Activity Diagram* Melihat Informasi Perangkat Sensor

Pada gambar 3.15 ini diperlihatkan, alur proses *Activity Diagram* melihat informasi perangkat sensor dimulai dengan

pengguna membuka halaman pengaturan perangkat sensor. Setelah itu, aplikasi akan menampilkan halaman pengaturan perangkat sensor dan pengguna dapat menekaan menu informasi perangkat sensor. Pengguna akan disajikan sebuah informasi tentang sensor seperti informasi pemeliharaan sensor, jenis sensor yang digunakan, serta penanganan sensor jika ada kendala.

14. *Activity Diagram* – Notifikasi (Perubahan Batasan Parameter Sensor)

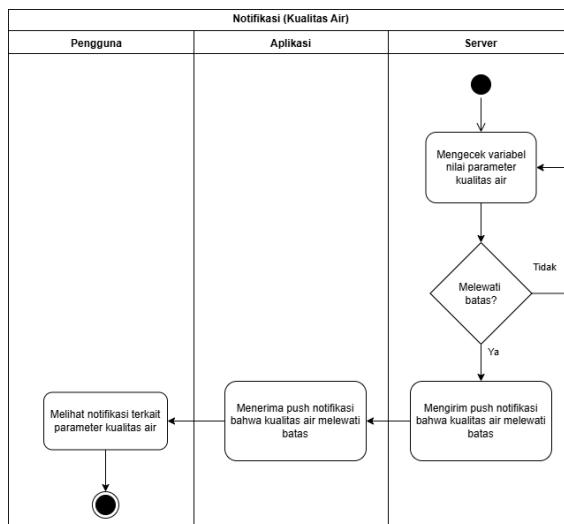


Gambar 3.16 *Activity Diagram* Notifikasi (Perubahan Batasan Parameter Sensor)

Pada gambar 3.16 ini diperlihatkan, alur proses *Activity Diagram* notifikasi jika ada perubahan pada batasan parameter sensor dimulai dari pengguna mengubah batasan parameter sensor. *Server* akan mengecek apakah ada perbuahan data batasan parameter sensor yang dilakukan pengguna. Jika tidak ada

perubahan data, maka *server* akan mengecek kembali *server* hingga ada perubahan data. Namun jika ada perubahan, *server* akan mengirimkan *push* notifikasi ke aplikasi bahwa ada batasan parameter sensor yang diubah. Sehingga pengguna dapat melihat notifikasi yang diberikan aplikasi.

15. Activity Diagram – Notifikasi (Kualitas Air)

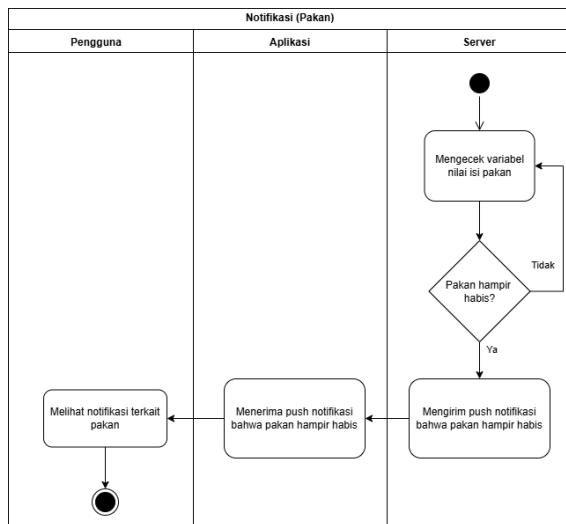


Gambar 3.17 *Activity Diagram* Notifikasi (Kualitas Air)

Pada gambar 3.17 ini diperlihatkan, alur proses *Activity Diagram* notifikasi jika ada kualitas air yang melewati batas toleransi dimulai dari *server* yang mengecek variabel nilai parameter kualitas air. Jika tidak ada parameter yang melewati batas, *server* akan kembali mengecek hingga terdapat parameter kualitas air yang melewati batas. Namun, jika ada parameter yang melewati batas, *server* akan mengirim *push* notifikasi ke aplikasi

bahwa kualitas air telah melewati batas toleransi. Sehingga pengguna dapat melihat notifikasi yang diberikan aplikasi.

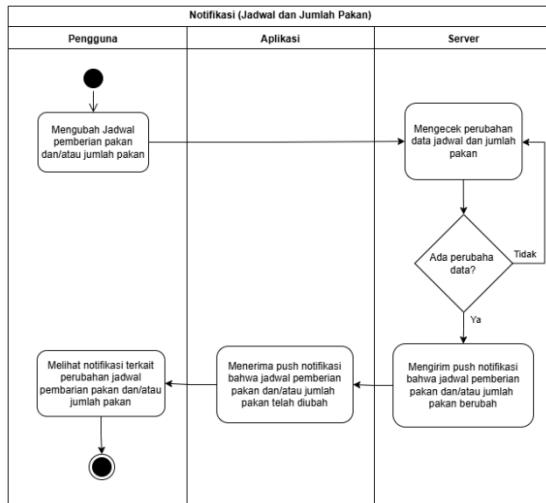
16. *Activity Diagram – Notifikasi (Pakan)*



Gambar 3.18 *Activity Diagram Notifikasi (Pakan)*

Pada gambar 3.18 ini diperlihatkan, alur proses *Activity Diagram* notifikasi jumlah pakan udang dalam tabung hampir habis dimulai dari *server* yang mengecek variabel nilai isi pakan. Jika pakan tidak habis, *server* akan kembali mengecek hingga variabel nilai isi pakan memenuhi kondisi. Namun, jika pakan hampir habis, *server* akan mengirim *push* notifikasi ke aplikasi bahwa pakan dalam tabung hampir habis. Sehingga pengguna dapat melihat notifikasi yang diberikan aplikasi.

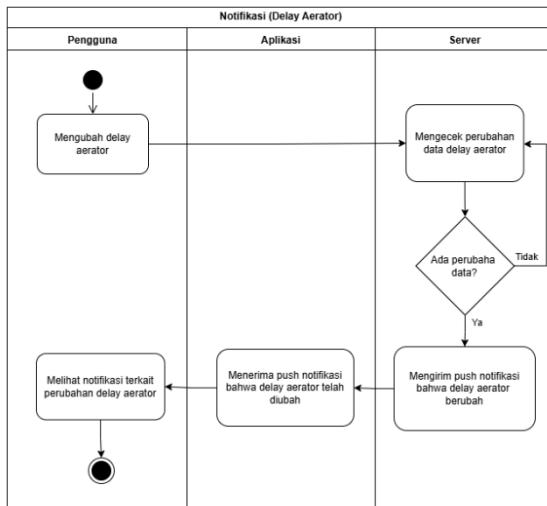
17. Activity Diagram – Notifikasi (Jadwal dan Jumlah Pakan)



Gambar 3.19 *Activity Diagram Notifikasi (Jadwal dan Jumlah Pakan)*

Pada gambar 3.19 ini diperlihatkan, alur proses *Activity Diagram* notifikasi jadwal pemberian dan jumlah pakan, proses ini dimulai dari *server* yang mengecek variabel nilai jadwal dan jumlah pakan. Jika jadwal dan jumlah pakan tidak berubah, *server* akan kembali mengecek hingga variabel nilai jadwal dan jumlah pakan memenuhi kondisi. Namun, jika jadwal dan jumlah pakan atau salah satunya berubah, *server* akan mengirim *push* notifikasi ke aplikasi bahwa jadwal dan jumlah pakan telah berubah. Sehingga pengguna dapat melihat notifikasi yang diberikan aplikasi.

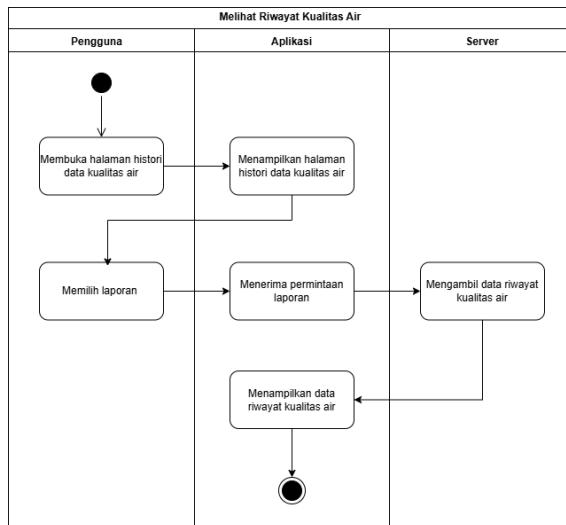
18. Activity Diagram – Notifikasi (Delay aerator)



Gambar 3.20 Activity Diagram Notifikasi (Delay Aerator)

Pada gambar 3.20 ini diperlihatkan, alur proses *Activity Diagram* notifikasi *Delay aerator*, proses ini dimulai dari *server* yang mengecek variabel nilai *Delay aerator*. Jika *Delay aerator* tidak berubah, *server* akan kembali mengecek hingga variabel nilai *Delay aerator* memenuhi kondisi. Namun, jika *Delay aerator* berubah, *server* akan mengirim *push* notifikasi ke aplikasi bahwa *Delay aerator* telah berubah. Sehingga pengguna dapat melihat notifikasi yang diberikan aplikasi.

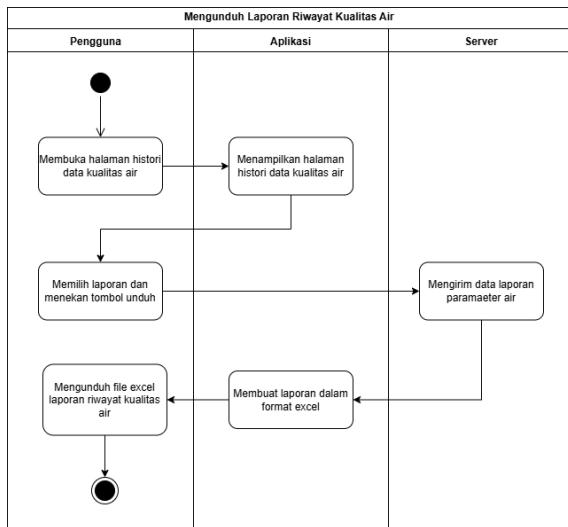
19. Activity Diagram – Melihat Riwayat Kualitas Air



Gambar 3.21 *Activity Diagram* Melihat Riwayat Kualitas Air

Pada gambar 3.21 ini diperlihatkan, alur proses *Activity Diagram* melihat riwayat kualitas air dimulai dari pengguna yang membuka halaman histori data kualitas air. Aplikasi akan menampilkan halaman histori data kualitas air dan memberikan daftar laporan yang di kumpulkan setiap hari. Laporan tersebut berisi riwayat data kualitas air tambak udang. Pengguna dapat memilih laporan yang disajikan oleh aplikasi, laporan yang dipilih akan diambil dari *server* terkait riwayat kualitas air. Kemudian, *server* mengirimkan data tersebut ke aplikasi untuk ditampilkan.

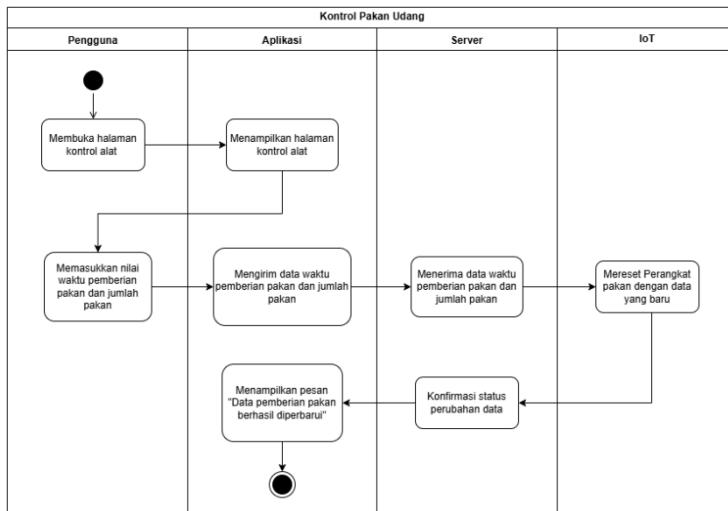
20. *Activity Diagram* – Mengunduh Laporan Riwayat Kualitas Air



Gambar 3.22 *Activity Diagram* Mengunduh Laporan Riwayat Kualitas Air

Pada gambar 3.22 ini diperlihatkan, alur proses *Activity Diagram* melihat mengunduh laporan riwayat kualitas air dimulai dari pengguna yang membuka halaman histori data kualitas air. Aplikasi akan menampilkan halaman histori data kualitas air dan memberikan daftar laporan. Pengguna dapat memiliki laporan yang disajikan oleh aplikasi, dan menekan tombol unduh untuk mengunduh laporan tersebut. Permintaan unduh laporan dari pengguna akan dikirimkan ke *server*. *Server* akan mengambil data terkait riwayat kualitas air dan mengirimkannya ke aplikasi untuk diolah menjadi laporan dengan format *excel*. Kemudian, pengguna dapat mengunduh laporan yang telah disediakan oleh aplikasi.

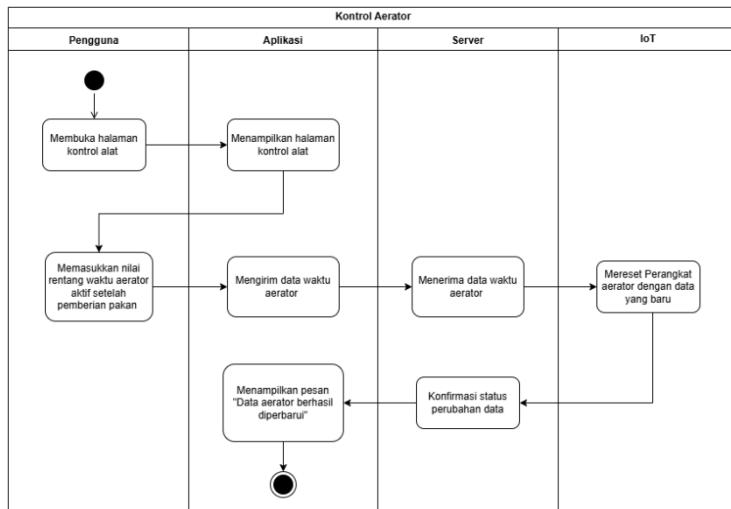
21. Activity Diagram – Kontrol Pakan Udang



Gambar 3.23 Activity Diagram Kontrol Pakan Udang

Pada gambar 3.23 ini diperlihatkan, alur proses *Activity Diagram* kontrol pakan udang dimulai dengan pengguna membuka halaman kontrol alat. Aplikasi akan menampilkan halaman kontrol alat yang terdapat tempat untuk mengukan nilai waktu pemberian pakan dan jumlah pakan yang ingin diberikan. Pengguna dapat mengukan data waktu dan jumlah pakan yang telah disediakan. *Input* pengguna akan dikirim ke *server* oleh aplikasi untuk di simpan. Kemudian, perangkat pakan udang akan *direset* sesuai dengan data waktu dan jumlah yang baru diberikan oleh pengguna. *Server* akan mengkonfirmasi status perubahan data, kemudian aplikasi akan menampilkan pesan bahwa data pemberian pakan berhasil diperbarui.

22. Activity Diagram – Kontrol Aerator



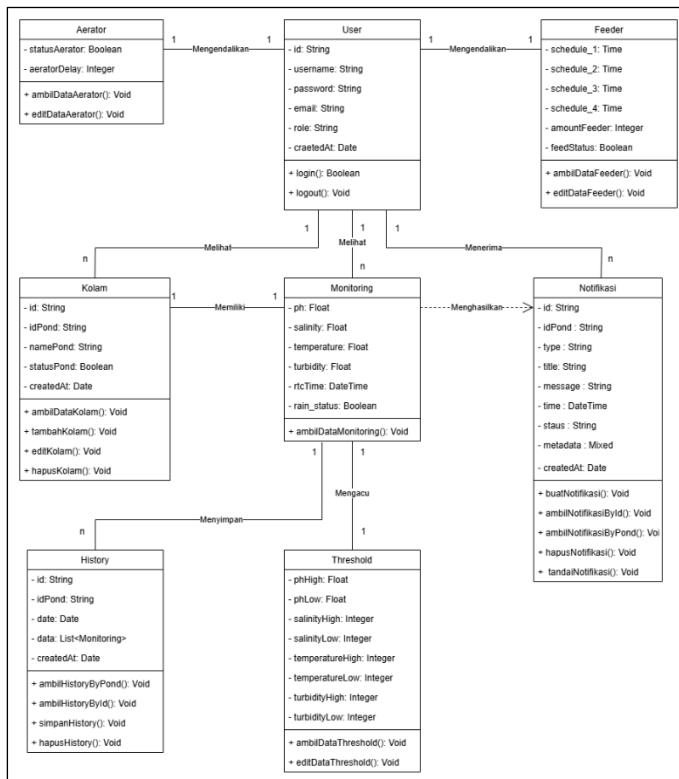
Gambar 3.24 *Activity Diagram* Kontrol Aerator

Pada gambar 3.24 ini diperlihatkan, alur *Activity Diagram* proses kontrol *aerator* dimulai dengan pengguna membuka halaman kontrol alat. Aplikasi akan menampilkan halaman kontrol alat yang terdapat tempat untuk menginputkan nilai rentang waktu *aerator* diaktifkan setelah pemberian pakan. Pengguna dapat menginputkan data waktu yang telah disediakan. Kemudian, *Server* akan mengkonfirmasi status perubahan data, kemudian aplikasi akan menampilkan pesan bahwa data pemberian pakan berhasil diperbarui

3.4.2.3 Class Diagram

Class Diagram menggambarkan struktur statis dari sistem dengan menampilkan kelas-kelas dalam sistem, atributnya, dan hubungan antar kelas. Ilustrasi di bawah ini menunjukkan kelas-kelas

yang terlibat dalam sistem, beserta atribut dan metode yang dimiliki masing-masing kelas serta hubungan di antara mereka.



Gambar 3.25 *Class Diagram*

Pada gambar 3.25 diperlihatkan *Class diagram* yang menggambarkan struktur sistem monitoring tambak udang dengan beberapa entitas utama yang saling berhubungan. Kelas *User* merupakan kelas dasar yang mencakup atribut umum seperti *id*, *username*, *password*, *email*, *role*, dan *createdAt*. *User* memiliki peran dalam memantau data kualitas air pada setiap kolam melalui kelas *Kolam* yang memiliki atribut *id*, *idPond*, *namePond*, *statusPond*, dan *createdAt*. Setiap *Kolam* memiliki perangkat sistem *Monitoring* yang

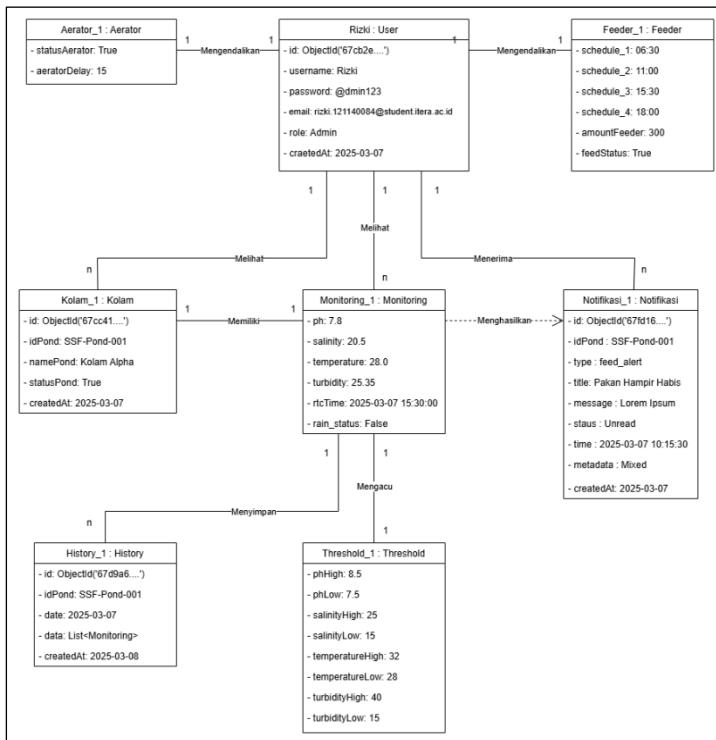
menunjukkan data sensor *real-time* seperti *ph*, *salinity*, *temperature*, *turbidity*, *rtcTime* dan *rain_status*.

Selain itu, Kelas Monitoring memiliki acuan parameter ambang batas kualitas air ditentukan dalam kelas *Threshold* yang terdiri dari atribut *pHHigh*, *pHLow*, *salinityHigh*, *salinityLow*, *temperatureHigh*, *temperatureLow*, *turbidityHigh*, dan *turbidityLow*. Kelas Notifikasi menyimpan informasi terkait notifikasi dengan atribut seperti *id*, *idPond*, *type*, *title*, *message*, *time*, *status*, dan *createdAt*. Sistem monitoring juga memiliki fitur *History*, yang menyimpan data historis dari pengukuran kualitas air tambak udang. Kelas *History* memiliki fungsi untuk menyimpan informasi terkait kondisi kualitas air pada waktu tertentu, dengan atribut seperti *id*, *idPond*, *date*, *data*, dan *createdAt*.

Selain itu, *User* juga dapat mengendalikan perangkat yang ada pada tambak udang, seperti *Aerator* dan *Feeder*. Kelas *Aerator* berfungsi untuk mengontrol perangkat *aerator* pada tambak, dengan atribut seperti *statusAerator* dan *aeratorDelay*. Kelas *Feeder* mengelola pemberian pakan secara otomatis pada tambak udang, dengan atribut seperti *schedule_1*, *schedule_2*, *schedule_3*, dan *schedule_4*, serta *amountFeeder* dan *feedStatus*. Melalui relasi antar kelas dalam *Class Diagram* sistem dirancang untuk memungkinkan pengguna memantau kualitas air tambak secara *real-time*, mengendalikan perangkat *aerator* dan *feeder*, membandingkan hasil pemantauan dengan batasan kualitas air, mengirimkan notifikasi saat terjadi anomali, serta menyimpan data historis untuk mendukung operasional dan pengelolaan tambak udang.

3.4.2.4 Object Diagram

Object Diagram menggambarkan struktur dinamis dari sistem dengan menampilkan objek-objek yang ada dalam sistem. *Diagram* ini memperlihatkan contoh spesifik dari instansi kelas yang ada dalam *Class Diagram*, termasuk nilai atribut yang dimiliki oleh objek-objek tersebut. Ilustrasi di bawah ini menunjukkan objek-objek yang terlibat dalam sistem, beserta atribut dan metode yang dimiliki masing-masing objek.



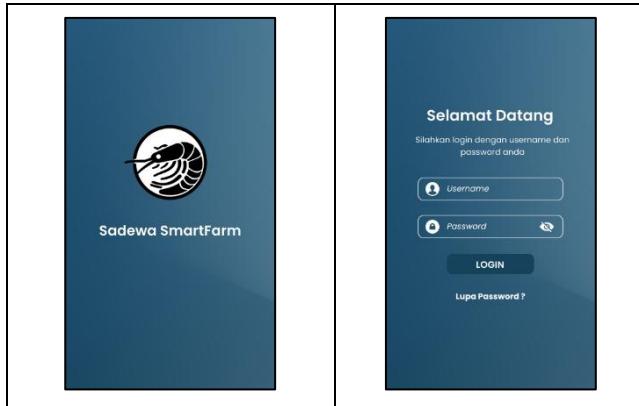
Gambar 3.26 *Object Diagram*

Gambar 3.26 merepresentasikan sebuah *Object Diagram* dari sistem monitoring tambak udang yang memperlihatkan instansiasi objek nyata dalam sistem serta hubungan antar entitas. Diagram ini

menampilkan objek User dengan atribut *username* "Rizki", *email* "rizki.121140084@student.itera.ac.id", role "Admin" dan tercatat pada 7 Maret 2025. Admin ini terkait dengan satu objek Kolam bernama Kolam Alpha dengan Id "SSF-Pond-001" dan status aktif. Kolam tersebut memiliki asosiasi ke dua objek perangkat, yaitu *Aerator* (status aktif, *delay* = 15 menit) dan *Feeder* (jadwal pemberian pakan pada pukul 06:30, 11:00, 15:30, dan 18:00, masing-masing 300 gram). Objek MonitoringKualitasAir mencatat parameter air waktu tertentu, yaitu pH sebesar 7.8, salinitas sebesar 20.5 *PPT*, suhu sebesar 28.0°C, dan kekeruhan sebesar 25.35 *NTU* pada 7 Maret 2025 pukul 15:30, dengan referensi terhadap nilai ambang batas normal, yaitu pH (7.5–8.5), salinitas (15–35 *PPT*), suhu (28–32°C), dan kekeruhan (15–40 *NTU*). Jika nilai pengamatan melebihi ambang batas, maka sistem akan membuat objek Notifikasi, seperti objek bertipe feed_alert berjudul "Pakan Hampir Habis", dikirimkan pada 7 Maret 2025 pukul 10:15:30 dengan status belum terbaca. Selain itu, objek RiwayatMonitoring menyimpan catatan historis kualitas air untuk dokumentasi dan analisis berkelanjutan.

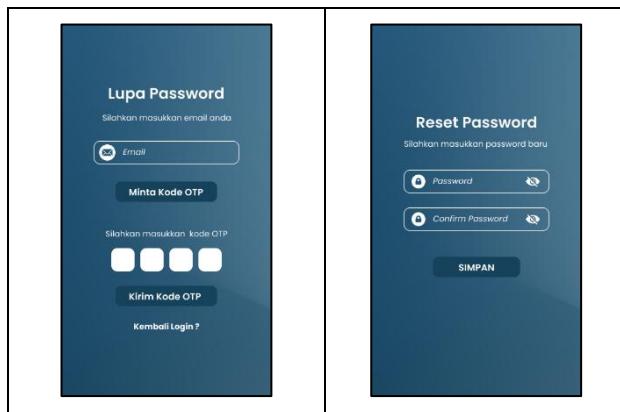
3.4.2.5 Tampilan antarmuka pengguna.

Desain antarmuka aplikasi yang interaktif dapat meningkatkan kemudahan penggunaan sistem oleh pengguna. Desain antarmuka aplikasi ini dibuat menggunakan *Figma*, alat bantu untuk pengembangan tampilan *UI* aplikasi. Desain antarmuka yang sudah dirancang akan diimplementasikan menjadi aplikasi yang sebenarnya. Rancangan aplikasi yang dibuat dalam penelitian ini yaitu.



Gambar 3.27 Halaman *Splash Screen* (Kiri) dan *Login* (Kanan)

Gambar 3.28 merupakan halaman *Splash Screen* yang berada di sebelah kiri dan halaman *Login* yang berada di sebelah kanan. Halaman *Splash Screen* akan muncul pertama kali ketika aplikasi dibuka. Halaman *Login* untuk memproses *autentikasi* pengguna agar dapat masuk ke aplikasi.



Gambar 3.28 Halaman *Lupa Password* (Kiri) dan *Reset Password* (Kanan)

Gambar 3.29 merupakan halaman *Lupa Password* yang berada di sebelah kiri dan *Reset Password* yang berada disebelah kanan. Halaman

Lupa Password berfungsi sebagai antarmuka untuk proses pemulihan kata sandi. Halaman *Reset Password* digunakan untuk membuat *password* baru.



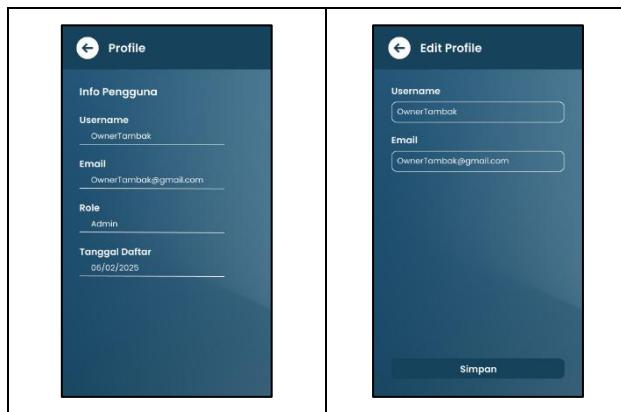
Gambar 3.29 Halaman Beranda Admin (Kiri) dan *User* (Kanan)

Gambar 3.30 menampilkan halaman Beranda Admin yang berada di sebelah kiri dan Beranda *User* yang berada di sebelah kanan. Halaman Beranda merupakan halaman pertama yang akan diakses oleh pengguna setelah *login*.



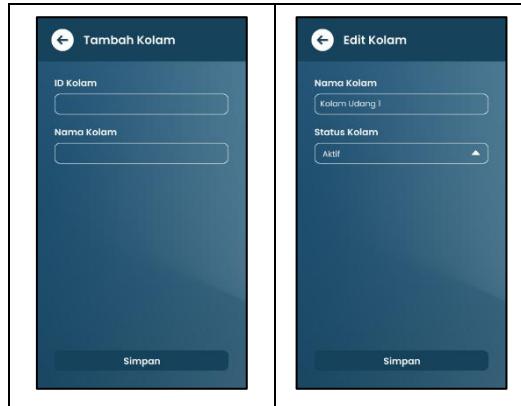
Gambar 3.30 *Popup* Menu Kolam (Kiri) dan Menu *Profile* (Kanan)

Gambar 3.31 merupakan tampilan *Popup Menu Kolam* yang berada disebelah kiri dan *Popup Menu Profile* yang berada disebelah kanan. Pada *Popup Menu Kolam* terdapat menu edit dan hapus kolam. Sedangkan, *Popup Menu Profile* terdapat menu untuk melihat info *profile*, edit *profile*, dan *logout*



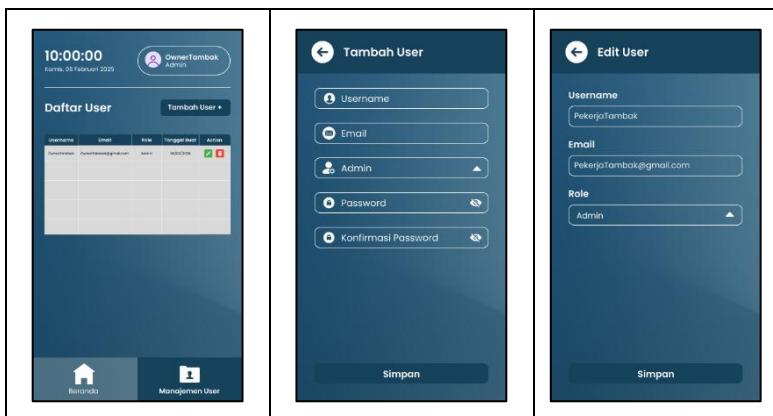
Gambar 3.31 Halaman *Profile* (Kiri) dan Edit *Profile* (Kanan)

Gambar 3.32 merupakan halaman *Profile* yang berada di sebelah kiri dan Edit *Profile* yang berada di sebelah kanan. Pada halaman *Profile*, pengguna dapat melihat informasi pribadi pengguna. Sedangkan, pada halaman Edit *Profile*, pengguna dapat memperbarui informasi pribadi yang digunakan pengguna.



Gambar 3.32 Halaman Tambah Kolam (Kiri) dan Edit Kolam (Kanan)

Gambar 3.33 merupakan halaman Tambah Kolam yang berada di sebelah kiri dan Edit Kolam yang berada di sebelah kanan. Kedua halaman ini berfungsi untuk menambahkan kolam dan mengedit kolam.



Gambar 3.33 Halaman Manajemen *User* (Kiri), Tambah *User* (Tengah) dan Edit *User* (Kanan)

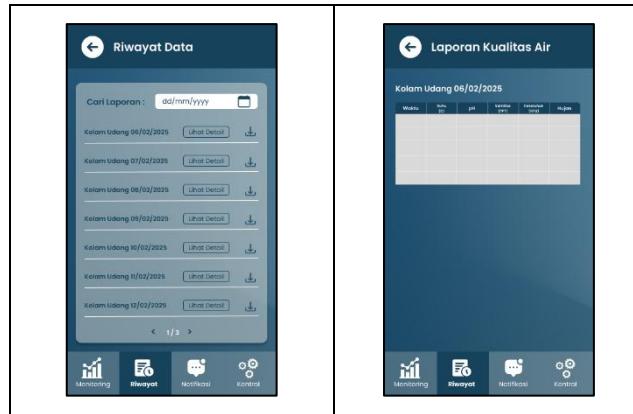
Gambar 3.34 merupakan halaman Manajemen *User* yang berada di sebelah kiri, Tambah *User* yang berada di tengah dan Edit *User* yang berada di sebelah kanan. Pada halaman Manajemen *User*, admin dapat

melihat daftar pengguna dan memanajemen pengguna. Pada halaman Tambah *User*, admin dapat menambah pengguna baru. Sedangkan, pada halaman Edit *User* admin dapat memperbarui data pengguna.



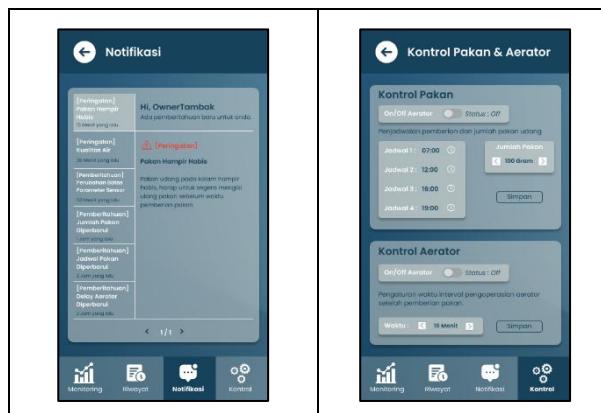
Gambar 3.34 Halaman Monitoring (Kiri), Pengaturan Batasan Sensor (Tengah) dan Informasi Sensor (Kanan)

Gambar 3.35 merupakan halaman Monitoring yang berada di sebelah kiri, halaman Pengaturan Sensor yang berada di sebelah tengah dan halaman Informasi Sensor yang berada di sebelah kanan. Pada halaman Monitoirng pengguna dapat melihat nilai kualitas air secara *real-time* dalam bentuk grafik. Pada halaman Pengaturan Batasan Sensor, pengguna dapat mengubah batasan parameter baik batasan tertinggi atau batasan terendah masing-masing sensor sesuai kebutuhan pemilik tambak. Sedangkan, pada halaman Informasi Sensor menyediakan informasi dasar tentang sensor, seperti tipe sensor yang digunakan, Deskripsi singkat sensor, spesifikasi sensor, dan batasan parameter yang direkomendasikan dari masing-masing sensor.



Gambar 3.35 Halaman Riwayat (Kiri) dan Laporan (Kanan)

Gambar 3.36 merupakan halaman Riwayat Data Monitoring yang berada di sebelah kiri dan halaman Laporan Data Monitoring yang berada di sebelah kanan. Pada halaman Riwayat Data Monitoring pengguna dapat melihat daftar riwayat kualitas air tambak perharinya. Sedangkan, pada halaman Laporan Data Monitoring akan menampilkan data kualitas air tambak.



Gambar 3.36 Halaman Notifikasi (Kiri) dan Kontrol (Kanan)

Gambar 3.37 merupakan tampilan halaman Notifikasi yang berada di sebelah kiri dan halaman Kontrol Pakan dan *Aerator* yang berada di sebelah kanan. Pada halaman Notifikasi pengguna dapat melihat notifikasi yang masuk ke aplikasi. Sedangkan, pada halaman Kontrol Pakan dan *Aerator* pengguna dapat mengatur jadwal pemberian pakan, jumlah pakan yang akan diberikan, *Delay aerator* menyala setelah pemberian pakan, dan pengguna dapat menghidupkan dan mematikan alat bila diperlukan.

3.4.3 Pengembangan Sistem (*Construction*)

Tahap Pengembangan Sistem melibatkan proses membangun sistem aplikasi berdasarkan desain yang telah disepakati dengan pengguna. Dalam tahap ini, pengembang bekerja cepat untuk mengimplementasikan spesifikasi dan rancangan yang telah dibuat. Selain itu, dilakukan juga integrasi aplikasi dengan sistem *Internet of Things (IoT)* agar perangkat fisik dapat terhubung dengan aplikasi. Data dari perangkat *IoT*, seperti sensor, dikirim ke sistem aplikasi untuk diolah dan ditampilkan secara *real-time*. Pengujian dilakukan secara bertahap untuk memastikan agar aplikasi bekerja dengan baik sesuai desain. Hasil dari pengembangan ini akan dibahas lebih detail pada Bab IV Hasil dan Pembahasan.

3.4.4 Implementasi (*Cutover*)

Pada tahap Implementasi, sistem yang telah dikembangkan dan diintegrasikan akan dioperasikan dalam lingkungan produksi. Pada tahap ini juga, akan dilakukan proses pengujian sistem menggunakan *Black Box Testing* dan *System Usability Scale (SUS)* untuk memastikan

bahwa sistem dapat berfungsi dengan baik dan memenuhi kebutuhan pengguna seperti yang diharapkan.

3.4.4.1 Black Box Testing

Pengujian *Black Box Testing* dilakukan untuk memverifikasi bahwa setiap fungsi aplikasi bekerja sesuai dengan spesifikasi yang telah ditentukan. Pengujian ini difokuskan pada *input* dan *output* aplikasi tanpa memperhatikan struktur internal atau kode program. Hasil dari pengujian ini akan menunjukkan apakah sistem telah memenuhi kebutuhan fungsional yang diharapkan. Berikut adalah pengujian untuk *Black Box Testing* dapat dilihat pada tabel 3.4.

Tabel 3.4 *Black Box Testing*

ID	Pengujian	Skenario Uji	Output yang Diharapkan
SK-01-1	<i>Login</i>	Pengguna memasukkan <i>email</i> dan <i>password</i> yang benar.	Aplikasi menampilkan pesan “ <i>Login berhasil!</i> ” dan pengguna diarahkan ke halaman Beranda jika memiliki <i>role Admin</i> dan ke halaman Beranda <i>User</i> jika memiliki <i>role User</i> .
SK-01-2		Pengguna memasukkan <i>username</i> dan/atau	Aplikasi menampilkan pesan <i>error</i> “ <i>Login gagal.</i> Periksa kembali <i>username/password.</i> ”

ID	Pengujian	Skenario Uji	Output yang Diharapkan
		<i>password</i> yang salah.	
SK-01-3		Pengguna memasukkan <i>username</i> dan <i>password</i> dengan salah satu <i>field</i> kosong.	Aplikasi menampilkan pesan <i>error</i> “ <i>Username</i> dan <i>password</i> tidak boleh kosong.”
SK-02-1	Lupa <i>Password</i>	Pengguna menekan tombol teks “Lupa <i>Password</i> ” pada halaman <i>login</i> .	Aplikasi menampilkan halaman Lupa <i>Password</i> .
SK-02-2		Pengguna memasukkan <i>email</i> yang terdaftar untuk menerima kode <i>OTP</i> .	Aplikasi menampilkan pesan “Kode OTP telah dikirim ke email Anda. Silakan cek email Anda.” dan pengguna menerima kode <i>OTP</i> .
SK-02-3		Pengguna memasukkan <i>email</i> yang tidak terdaftar.	Aplikasi menampilkan pesan <i>error</i> “Gagal mengirim <i>OTP</i> . Pastikan <i>email</i> sudah terdaftar.”

ID	Pengujian	Skenario Uji	Output yang Diharapkan
SK-02-4		Pengguna memasukkan kode <i>OTP</i> yang benar.	Pengguna berhasil masuk ke halaman <i>Reset Password</i> .
SK-02-5		Pengguna memasukkan kode <i>OTP</i> yang salah atau kadaluarsa.	Pengguna gagal masuk ke halaman <i>Reset Password</i> dan Aplikasi menampilkan pesan <i>error</i> .
SK-02-6		Pengguna tidak memasukkan <i>email</i> atau kode <i>OTP</i> dan langsung klik tombol.	Aplikasi menampilkan pesan <i>error</i> .
SK-03-1	<i>Reset Password</i>	Pengguna memasukkan <i>password</i> baru dan <i>confirm password</i> yang valid.	Aplikasi menampilkan pesan “ <i>Password</i> berhasil diperbarui. Silakan <i>login</i> dengan <i>password</i> baru.” dan pengguna kembali ke halaman <i>login</i> .
SK-03-2		Pengguna memasukkan <i>password</i> baru dan/atau <i>confirm</i>	Aplikasi menampilkan pesan <i>error</i> .

ID	Pengujian	Skenario Uji	Output yang Diharapkan
		<i>password</i> dengan data yang tidak valid atau kosong.	
SK-04-1	Halaman Beranda	Pengguna <i>login</i> menggunakan akun Admin.	Aplikasi menampilkan halaman beranda dengan fitur-fitur dan menu yang lengkap, termasuk Manajemen <i>User</i> dan Manajemen Kolam.
SK-04-2		Pengguna <i>login</i> menggunakan akun <i>User</i> .	Aplikasi menampilkan halaman beranda tanpa adanya menu Manajemen <i>User</i> dan Manajemen Kolam.
SK-05-1	<i>Info Profile</i>	Pengguna memilih opsi “Info Profile” dari <i>dropdown</i> menu pada <i>widget profile</i> .	Halaman <i>Profile</i> pengguna berhasil diakses dan menampilkan data pengguna yang sesuai.
SK-06-1	<i>Edit Profile</i>	Pengguna memilih opsi “Edit Profile” dari <i>dropdown</i>	Aplikasi menampilkan halaman edit <i>profile</i> dengan data <i>profile</i> yang telah terisi di setiap <i>field</i> .

ID	Pengujian	Skenario Uji	Output yang Diharapkan
		menu pada <i>widget profile</i> .	
SK-06-2		Pengguna memperbarui <i>username</i> dan <i>email</i> dengan data yang valid.	Aplikasi menampilkan pesan “Profile berhasil diperbarui!” dan diarahkan ke halaman <i>Profile</i> .
SK-06-3		Pengguna memperbarui <i>username</i> dan/atau <i>email</i> dengan data yang tidak valid atau kosong.	Aplikasi menampilkan pesan <i>error</i> .
SK-07-1	Status Notifikasi	Pengguna memilih opsi “Notifikasi” dari <i>dropdown menu</i> pada <i>widget profile</i> , lalu menekan <i>toggle</i> ke posisi On.	Aplikasi menampilkan pesan "Notifikasi diaktifkan".
SK-07-2		Pengguna memilih opsi “Notifikasi” dari	Aplikasi menampilkan dialog konfirmasi untuk

ID	Pengujian	Skenario Uji	Output yang Diharapkan
		<i>dropdown menu pada widget profile</i> , lalu menekan <i>toggle</i> ke posisi Off.	mematikan <i>popup notifikasi</i> .
SK-07-3		Pengguna menekan tombol "Ya, Matikan" pada dialog konfirmasi.	Status notifikasi berhasil dinonaktifkan, dan <i>switch</i> berubah ke posisi Off.
SK-07-4		Pengguna menekan tombol "Batal" pada dialog konfirmasi.	Status notifikasi tetap aktif, dan <i>switch</i> kembali ke posisi On.
SK-08-1	<i>Logout</i>	Pengguna memilih opsi “Logout” dari <i>dropdown menu pada widget profile</i> .	Aplikasi menampilkan dialog konfirmasi <i>logout</i> .
SK-08-2		Pengguna menekan tombol "Ya" pada	Aplikasi melakukan proses <i>logout</i> dan kembali ke halaman <i>login</i> .

ID	Pengujian	Skenario Uji	Output yang Diharapkan
		dialog konfirmasi.	
SK-08-3		Pengguna menekan tombol "Batal" pada dialog konfirmasi.	Proses <i>logout</i> dibatalkan, dan pengguna tetap berada di halaman sebelumnya.
SK-09-1	Halaman Manajemen <i>User</i>	Admin memilih menu “Manajemen <i>User</i> ” pada bilah navigasi di halaman beranda.	Aplikasi menampilkan halaman Manajemn <i>User</i> beserta daftar pengguna aplikasi dan fitur manajemen <i>user</i> seperti tambah, hapus dan edit.
SK-10-1	Tambah Pengguna Aplikasi	Pada halaman Manajemen <i>User</i> , Admin menekan tombol “Tambah <i>User</i> ”.	Aplikasi menampilkan halaman Tambah <i>User</i> .
SK-10-2		Admin memasukkan data pengguna baru seperti <i>username</i> , <i>email</i> , <i>role</i> ,	Aplikasi menampilkan pesan “ <i>User</i> berhasil ditambahkan” dan pengguna baru berhasil ditambahkan ke dalam aplikasi dan muncul

ID	Pengujian	Skenario Uji	Output yang Diharapkan
		<p><i>password</i> dan <i>confirm password</i> dengan data yang valid.</p>	dalam daftar pengguna aplikasi.
SK-10-3		<p>Admin memasukkan data pengguna baru seperti <i>username</i>, <i>email</i>, <i>role</i>, <i>password</i> dan/atau <i>confirm password</i> dengan data yang tidak valid atau kosong.</p>	Aplikasi menampilkan pesan <i>error</i> .
SK-11-1	Edit Pengguna Aplikasi	<p>Pada halaman Manajemen <i>User</i>, Admin memilih pengguna dari daftar, lalu menekan tombol ikon tiga titik di</p>	Aplikasi menampilkan halaman Edit Pengguna dengan data Pengguna yang telah terisi di setiap <i>fieldnya</i> .

ID	Pengujian	Skenario Uji	Output yang Diharapkan
		pojok kanan atas <i>tile user</i> , kemudian pada <i>dropdown</i> yang muncul, pengguna memilih menu edit.	
SK- 11-2		Admin memperbarui <i>username</i> , <i>email</i> , dan <i>role</i> <i>user</i> dengan data yang valid	Aplikasi menampilkan pesan “Data pengguna berhasil diperbarui!”
SK- 11-3		Admin memperbarui <i>username</i> , <i>email</i> , dan/atau <i>role user</i> dengan data yang tidak valid atau kosong.	Aplikasi menampilkan pesan <i>error</i> .
SK- 12-1	Hapus Pengguna Aplikasi	Pada halaman Manajemen <i>User</i> , Admin	Aplikasi menampilkan dialog konfirmasi penghapusan pengguna.

ID	Pengujian	Skenario Uji	Output yang Diharapkan
		<p>memilih pengguna dari daftar, lalu menekan tombol ikon tiga titik di pojok kanan atas <i>tile user</i>,</p> <p>kemudian pada <i>dropdown</i> yang muncul,</p> <p>pengguna memilih menu hapus.</p>	
SK-12-2		Admin menekan tombol "Hapus" pada dialog konfirmasi.	Pengguna berhasil dihapus dan tidak muncul lagi dalam tabel daftar pengguna.
SK-12-3		Admin menekan tombol "Batal" pada dialog konfirmasi.	Proses penghapusan dibatalkan, dan pengguna tetap ada dalam tabel daftar pengguna.
SK-13-1	Tambah Kolam Baru	Pada halaman beranda, admin menekan tombol "Tambah Kolam"	Aplikasi menampilkan halaman tambah kolam.

ID	Pengujian	Skenario Uji	Output yang Diharapkan
SK-13-2		Admin memasukkan data <i>id</i> kolam dan nama kolam dengan data yang valid.	Aplikasi menampilkan pesan “Kolam berhasil ditambahkan.”
SK-13-3		Admin memasukkan data <i>id</i> kolam dan/atau nama kolam dengan data yang tidak valid atau kosong.	Aplikasi menampilkan pesan <i>error</i> .
SK-14-1	Edit Kolam	Pada halaman beranda, Admin menekan ikon tiga titik di pojok kanan atas pada <i>tile</i> kolam, lalu memilih opsi "Edit" dari <i>dropdown</i> menu.	Aplikasi menampilkan halaman Edit Kolam dengan data Pengguna yang telah terisi di setiap <i>fieldnya</i> .

ID	Pengujian	Skenario Uji	Output yang Diharapkan
SK-14-2		Admin memperbarui data nama kolam dan status kolam dengan data yang valid.	Aplikasi menampilkan pesan “Kolam berhasil diperbarui!”
SK-14-3		Admin memperbarui data nama kolam dengan data yang tidak valid atau kosong.	Aplikasi menampilkan pesan <i>error</i> .
SK-14-4		Admin memperbarui status kolam menjadi Aktif/Non-Aktif.	Aplikasi menampilkan pesan “Kolam berhasil diperbarui!”
SK-15-1	Hapus Kolam	Pada halaman beranda, Admin menekan ikon tiga titik di pojok kanan atas pada <i>tile</i> kolam,	Aplikasi menampilkan dialog konfirmasi penghapusan kolam.

ID	Pengujian	Skenario Uji	Output yang Diharapkan
	SK-15-2	lalu memilih opsi "Hapus" dari <i>dropdown</i> menu.	
SK-15-2		Admin menekan tombol "Hapus" pada dialog konfirmasi.	Kolam berhasil dihapus dan tidak muncul lagi dalam daftar kolam.
SK-15-3		Admin menekan tombol "Batal" pada dialog konfirmasi.	Proses penghapusan dibatalkan, dan kolam tetap ada dalam daftar.
SK-16-1	Halaman Monitoring	Pada halaman beranda, pengguna memilih kolam dari daftar kolam dengan status kolam Aktif, lalu menekan tombol “Monitoring” pada <i>tile</i> kolam yang dipilih.	Aplikasi menampilkan halaman Monitoring beserta <i>tile</i> grafik sensor (suhu, ph, salinitas, dan kekeruhan).

ID	Pengujian	Skenario Uji	Output yang Diharapkan
SK-16-2		<p>Pada halaman beranda, pengguna memilih kolam dari daftar kolam dengan status kolam Non-Aktif, lalu menekan tombol “Monitoring” pada <i>tile</i> kolam yang dipilih.</p>	Aplikasi menampilkan pesan <i>error</i> “Monitoring untuk kolam ini dinonaktifkan karena status kolam adalah Non-Aktif.”
SK-17-1	Menampilkan Nilai Kualitas Air	Aplikasi mengambil data sensor kualitas air (suhu, pH, salinitas, kekeruhan) secara <i>real-time</i> .	Aplikasi menampilkan parameter kualitas air secara <i>real-time</i> (suhu, pH, salinitas, kekeruhan).
SK-18-1	Grafik Kualitas Air	Aplikasi mengambil data kualitas air (suhu, pH, salinitas, kekeruhan)	Aplikasi menampilkan grafik yang memperbarui perubahan parameter kualitas air (suhu, pH, salinitas, kekeruhan) setiap 10 detik.

ID	Pengujian	Skenario Uji	Output yang Diharapkan
		setiap 10 detik, lalu memperbarui tampilan dalam bentuk grafik.	
SK- 19-1	Halaman Pengaturan	Pada halaman monitoring, pengguna memilih <i>tile</i> grafik sensor kualitas air yang dipilih, lalu menekan tombol teks “Pengaturan Sensor <Nama_Sensor> ”.	Aplikasi menampilkan halaman Pengaturan yang berisi <i>input</i> untuk mengubah nilai batas minimum dan maksimum sensor yang dipilih.
SK- 20-1	Mengubah Batasan Parameter Sensor	Pengguna memasukkan nilai batas tertinggi dan/atau terendah	Aplikasi menampilkan pesan "Batasan sensor <Nama Sensor> berhasil diperbarui."

ID	Pengujian	Skenario Uji	Output yang Diharapkan
SK-20-2		parameter sensor.	
		Pengguna memasukkan nilai batas tertinggi lebih rendah dari batas terendah parameter sensor.	Aplikasi menampilkan pesan “Batas bawah harus lebih kecil dari batas atas.”
SK-21-1	Halaman Informasi Sensor	Pada halaman pengaturan sensor, pengguna menekan tombol teks "Informasi Perangkat".	Aplikasi menampilkan informasi perangkat sensor yang sesuai dengan sensor yang dipilih, termasuk nama, gambar, deskripsi, spesifikasi, dan rentang optimal penggunaan.
SK-22-1	Halaman Notifikasi	Pengguna memilih menu “Notifikasi” pada bilah navigasi di halaman monitoring.	Aplikasi menampilkan halaman Notifikasi yang memuat daftar notifikasi yang relevan dengan kolom yang dipilih di halaman Beranda.

ID	Pengujian	Skenario Uji	Output yang Diharapkan
SK-23-1	Notifikasi Parameter Melewati Batas Aman	Aplikasi memeriksa perubahan nilai sensor pada air kolam dari sistem.	Aplikasi mengirimkan pesan notifikasi ‘Kualitas <Nama Sensor> air pada <Nama Kolam> berada di luar batas normal. <Nama Sensor> = <Nilai Sensor>.’
		Aplikasi memeriksa perubahan nilai kualitas air kolam lebih dari satu secara bersamaan dari sistem.	Aplikasi mengirimkan pesan notifikasi ‘Kualitas <Sensor1, Sensor2, ...> air pada <Nama Kolam> berada di luar batas normal. Sensor1 = <Nilai Sensor1>. Sensor2 = <Nilai Sensor2>.’
SK-24-1	Notifikasi Perubahan Batasan Parameter Sensor	Aplikasi memeriksa perubahan nilai batasan maksimum dan minimum parameter sensor.	Aplikasi mengirimkan pesan notifikasi ”Batas Parameter pada <Nama Kolam> telah diubah : <Nama Sensor> : <Batasan Lama> => <Batasan Baru>.”
SK-25-1	Notifikasi Perubahan	Aplikasi memeriksa	Aplikasi mengirimkan pesan notifikasi

ID	Pengujian	Skenario Uji	Output yang Diharapkan
SK-25	Data Jadwal Pemberian Pakan	perubahan jadwal pemberian pakan.	“Konfigurasi pakan pada <Nama Kolam> telah diperbarui: Jadwal: <Jadwal Lama> => <Jadwal Baru>.”
		Aplikasi memeriksa perubahan jumlah pakan.	Aplikasi mengirimkan pesan notifikasi “Konfigurasi pakan pada <Nama Kolam> telah diperbarui: Jumlah: <Jumlah Lama> => <Jumlah Baru> gram.”
		Aplikasi menerima perubahan jumlah pakan dan jadwal pemberian pakan secara bersamaan.	Aplikasi mengirimkan pesan notifikasi “Konfigurasi pakan pada <Nama Kolam> telah diperbarui: Jadwal: <Jadwal Lama> => <Jadwal Baru>. Jumlah: <Jumlah Lama> => <Jumlah Baru> gram.”
SK-26-1	Notifikasi Perubahan	Aplikasi memeriksa	Aplikasi mengirimkan pesan notifikasi “ <i>Delay aerator</i> untuk <Nama

ID	Pengujian	Skenario Uji	Output yang Diharapkan
	<i>Delay Aerator</i>	perubahan <i>Delay aerator.</i>	Kolam> telah diubah dari < <i>Delay Lama</i> > menjadi < <i>Delay Baru</i> >.”
SK-27-1	Notifikasi Jumlah Pakan Hampir Habis	Aplikasi memeriksa isi pakan udang di dalam tabung pakan.	Aplikasi mengirimkan pesan notifikasi “Pakan pada kolam Kolam Beta hampir habis, harap segera isi ulang.”
SK-28-1	Melihat Notifikasi	Pada halaman notifikasi, pengguna memilih notifikasi yang ingin dilihat di daftar notifikasi.	Notifikasi yang dipilih ditampilkan dengan detail lengkap dan warna judul notifikasi berubah menjadi putih sebagai tanda telah dibaca.
SK-29-1	Hapus Notifikasi	Aplikasi menghapus notifikasi yang telah berumur lebih dari 7 hari.	Notifikasi yang telah berumur lebih dari 7 hari tidak muncul di dalam daftar notifikasi.
SK-30-1	Halaman Riwayat Kualitas air	Pengguna memilih menu “Riwayat” pada bilah navigasi di	Aplikasi menampilkan halaman riwayat kualitas air yang berisi daftar laporan kualitas air.

ID	Pengujian	Skenario Uji	Output yang Diharapkan
		halaman monitoring.	
SK-31-1	Pencarian Laporan Riwayat Kualitas Air	Pengguna memasukkan tanggal pada kolom <i>input</i> pencarian, dimana tanggal yang <i>diinput</i> memiliki laporan yang sesuai di daftar laporan.	Aplikasi menampilkan laporan kualitas air sesuai dengan tanggal yang dimasukkan.
SK-31-2		Pengguna memasukkan tanggal pada kolom <i>input</i> pencarian, dimana tanggal yang <i>diinput</i> tidak memiliki laporan di daftar laporan.	Aplikasi menampilkan pesan "Laporan Riwayat Kualitas Air pada Tanggal <Tanggal Laporan> tidak tersedia."
SK-31-3		Pengguna menekan icon	Aplikasi menghapus <i>input</i> tanggal pencarian dan

ID	Pengujian	Skenario Uji	Output yang Diharapkan
		"X" di dalam kolom pencarian, setelah melakukan pencarian.	menampilkan kembali seluruh daftar laporan.
SK-32-1	Melihat Laporan Riwayat Kualitas Air	Pada halaman riwayat, pengguna menekan tombol “Lihat Detail” pada riwayat yg dipilih.	Aplikasi menampilkan halaman Laporan Kualitas Air.
SK-33-1	Mengunduh Laporan Riwayat Kualitas Air	Pada halaman riwayat, pengguna menekan ikon unduh pada riwayat yg dipilih.	Aplikasi menampilkan pesan “Laporan berhasil disimpan. Lokasi file: <Lokasi File>”
SK-34-1	Hapus Laporan Riwayat Kualitas Air	Aplikasi menghapus laporan riwayat kualitas air yang telah berumur	Laporan riwayat kualitas air yang telah berumur lebih dari 30 hari tidak muncul di dalam daftar laporan.

ID	Pengujian	Skenario Uji	Output yang Diharapkan
		lebih dari 30 hari.	
SK-35-1	Halaman Kontrol Pakan dan <i>Aerator</i>	Pengguna memilih menu “Kontrol” pada bilah navigasi di halaman monitoring.	Aplikasi menampilkan halaman kontrol pakan dan <i>aerator</i> .
SK-36-1	Kontrol Otomatis Pelontar Pakan dan <i>Aerator</i>	Aplikasi memeriksa jadwal pakan. Jika waktu saat ini sesuai dengan jadwal pakan, aplikasi mengaktifkan pelontar pakan.	Pelontar pakan bekerja secara otomatis pada waktu yang telah ditentukan.
SK-36-2		Aplikasi memeriksa nilai <i>Delay aerator</i> . Jika pakan telah diberikan, aplikasi menunda aktivasi <i>aerator</i>	<i>Aerator</i> berhenti otomatis saat pemberian pakan dilakukan. <i>Delay</i> sesuai dengan <i>Delay aerator</i> telah ditentukan.

ID	Pengujian	Skenario Uji	Output yang Diharapkan
		berdasarkan nilai <i>Delay</i> yang telah ditentukan.	
SK-37-1	Kontrol Manual Pelontar Pakan dan <i>Aerator</i>	Pengguna menekan tombol "On" pada <i>tile</i> kontrol pakan.	Aplikasi menampilkan pesan "Pakan berhasil diaktifkan" dan pelontar pakan mulai bekerja.
SK-37-2		Pengguna menekan tombol "Off" pada <i>tile</i> kontrol pakan.	Aplikasi menampilkan pesan "Pakan berhasil dinonaktifkan" dan pelontar pakan mulai berhenti.
SK-37-3		Pengguna menekan tombol "On" pada <i>tile</i> kontrol <i>aerator</i> .	Aplikasi menampilkan pesan " <i>Aerator</i> berhasil diaktifkan" dan <i>aerator</i> mulai bekerja.
SK-37-4		Pengguna menekan tombol "Off" pada <i>tile</i> kontrol <i>aerator</i> .	Aplikasi menampilkan pesan " <i>Aerator</i> berhasil dinonaktifkan" dan <i>aerator</i> mulai berhenti.
SK-38-1	Pengaturan Jadwal Pemberian Pakan.	Pengguna memasukkan waktu yang diinginkan	Aplikasi menampilkan pesan "Konfigurasi berhasil disimpan" dan menyimpan jadwal

ID	Pengujian	Skenario Uji	Output yang Diharapkan
	SK-38-2	untuk pemberian pakan.	pemberian pakan yang baru.
SK-38-2		Pengguna memilih jumlah pakan yang diinginkan	Aplikasi menampilkan pesan “Konfigurasi berhasil disimpan” dan menyimpan jumlah pakan udang yang baru.
SK-38-3		Pengguna memasukkan waktu dan jumlah pakan secara bersamaan.	Aplikasi menampilkan pesan “Konfigurasi berhasil disimpan” dan menyimpan waktu dan jumlah pakan udang yang baru.
SK-39-1	<i>Delay Aerator</i>	Pengguna memilih nilai <i>Delay</i> (waktu tunda) untuk <i>aerator</i> setelah pemberian pakan.	Aplikasi menampilkan pesan “Waktu <i>Delay aerator</i> berhasil diperbarui.”

3.4.4.2 System *Usability* Scale

Pengujian *System Usability Scale (SUS)* digunakan untuk mengukur tingkat kegunaan (*usability*) dari aplikasi yang telah dikembangkan. Pengujian *SUS* dipilih karena metode ini sederhana

namun efektif dalam mendapatkan umpan balik tentang pengalaman pengguna saat menggunakan aplikasi. Pada penelitian ini, sebanyak 30 responden yang merupakan anggota kelompok budidaya Sadewa Farm dilibatkan untuk mengisi kuesioner *SUS*. Jumlah tersebut dipilih karena telah memenuhi batas minimum yang disarankan dalam penggunaan metode *SUS*, yaitu 30 responden, yang dianggap memadai untuk menghasilkan data uji yang valid dan mendekati distribusi normal.

Proses pengumpulan data dilakukan dengan menyusun kuesioner *SUS* dalam bentuk *Google Form (GForm)* dan dibagikan secara daring kepada para responden. Masing-masing responden diminta memberikan penilaian terhadap 10 pernyataan *SUS* menggunakan skala Likert 1 hingga 5, di mana skor 1 menunjukkan ketidaksepakatan yang sangat kuat dan skor 5 menunjukkan persetujuan yang sangat kuat. Responden memberikan penilaian yang mencakup aspek seperti kemudahan penggunaan, kejelasan antarmuka, dan kepuasan keseluruhan. Hasil pengujian dapat dilihat pada Tabel 3.5.

Tabel 3.5 Pertanyaan - Pertanyaan System *Usability Scale*

No.	Pertanyaan	1	2	3	4	5
1.	Saya berfikir akan menggunakan sistem ini lagi.					
2.	Saya merasa sistem ini rumit untuk digunakan.					
3.	Saya merasa sistem ini mudah untuk digunakan.					

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

BAB IV

HASIL DAN PEMBAHASAN

4.1. Hasil Penelitian

Penelitian ini menerapkan metode *Rapid Application Development (RAD)* untuk mengembangkan aplikasi monitoring dan kontrol otomatis pada budidaya udang. Metode *RAD* dipilih karena kemampuannya dalam menghasilkan solusi teknologi secara cepat dan iteratif, dengan tahapan utama yang meliputi perencanaan kebutuhan, desain pengguna, pengembangan sistem, dan implementasi. Pada tahap perencanaan kebutuhan, dilakukan identifikasi permasalahan melalui wawancara dengan salah satu pemilik tambak udang, yaitu Sadewa Farm. Hasil wawancara tersebut dianalisis dan dirumuskan menjadi kebutuhan sistem, baik fungsional maupun non-fungsional.

4.1.1. Desain Pengguna (*Design User*)

Setelah tahap perencanaan kebutuhan diselesaikan, proses pengembangan berlanjut ke tahap desain pengguna (*User Design*). Pada tahap ini, dilakukan pembuatan prototipe awal dari aplikasi. Prototipe ini kemudian diuji secara langsung oleh pengguna dan hasil dari pengujian ini digunakan untuk melakukan evaluasi dan perbaikan (*refine*) terhadap desain agar semakin sesuai dengan ekspektasi pengguna. Proses ini dilakukan secara berulang melalui beberapa iterasi hingga desain antarmuka mendapatkan persetujuan dari pihak terkait.

4.1.1.1. Iterasi Pertama

A. Prototype

Gambar 4.1 berikut menampilkan tampilan awal halaman Manajemen *User*, yang hanya dapat diakses oleh admin untuk mengelola pengguna aplikasi, seperti menambahkan pengguna baru, mengedit pengguna dari daftar yang sudah ada, dan menghapus pengguna dari daftar. Tampilan awal halaman ini menggunakan format tabel untuk menyajikan daftar pengguna.



Gambar 4.1 Halaman Manajemen *User* Sebelum Perbaikan

B. Test

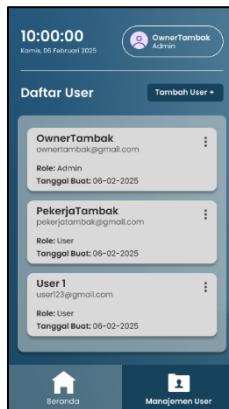
Hasil evaluasi pada iterasi pertama menghasilkan beberapa saran dan masukan dari pihak tambak udang Sadewa Farm, yang menjadi pengguna utama aplikasi ini. Evaluasi dilakukan terhadap keterbacaan dan kemudahan interaksi pada tampilan aplikasi, terutama halaman Manajemen *User*. Berikut adalah tabel evaluasi iterasi pertama yang akan ditampilkan pada Tabel 4.1 dibawah ini.

Tabel 4.1 Evaluasi Desain Iterasi Pertama

Fitur	Masukan dan Saran
Halaman Manajemen User	Pihak Tambak Udang Sadewa Farm menyarankan agar tampilan daftar pengguna tidak lagi menggunakan format tabel karena teks terlalu kecil dan sulit dibaca di perangkat mobile. Mereka menyarankan untuk mengubah tampilan agar lebih mudah dibaca.

C. Refine

Berdasarkan hasil evaluasi dan masukan yang diterima pada tahap *Test*, dilakukan perbaikan pada desain halaman Manajemen *User*. Perubahan yang dilakukan adalah mengganti tampilan daftar *user* dari tabel menjadi *card layout* agar informasi yang disajikan menjadi lebih proporsional dan mudah terbaca di perangkat *mobile*. Tampilan hasil perbaikan Halaman Manajemen *User* akan ditampilkan pada Gambar 4.2 berikut.

Gambar 4.2 Halaman Manajemen *User* Setelah Perbaikan

4.1.1.2. Iterasi Kedua

A. Prototype

Gambar 4.3 berikut menampilkan tampilan awal dari *popup* menu *profile* yang menampilkan beberapa opsi seperti *Info Profile*, *Edit Profile*, dan *Logout*. Tampilan ini menjadi dasar dalam pengujian dan penyempurnaan fitur pada iterasi kedua.



Gambar 4.3 *Popup* Menu *Profile* Sebelum Perbaikan

B. Test

Hasil evaluasi pada iterasi kedua menghasilkan beberapa saran dan masukan dari pihak tambak udang Sadewa Farm, yang menjadi pengguna utama aplikasi ini. Evaluasi dilakukan terhadap kenyamanan dan kontrol pengguna dalam menggunakan aplikasi, khususnya terkait pengaturan personal. Berikut adalah tabel evaluasi iterasi kedua yang akan ditampilkan pada Tabel 4.2 dibawah ini.

Tabel 4.2 Evaluasi Desain Iterasi Kedua

Fitur	Masukan dan Saran
<i>Popup Menu Profile</i>	Pihak Tambak Udang Sadewa Farm memberikan masukan untuk menyediakan kontrol untuk mengatur status notifikasi. Beberapa pengguna merasa terganggu oleh notifikasi yang muncul saat penggunaan aplikasi, terutama ketika sedang bekerja.

C. Refine

Berdasarkan hasil evaluasi dan masukan yang diterima pada tahap *Test*, dilakukan perbaikan pada desain *Popup Menu Profile* dengan menambahkan fitur berupa tombol *switch* yang memungkinkan pengguna mengelola status notifikasi. Tampilan hasil perbaikan *Popup Menu Profile* akan ditampilkan pada Gambar 4.4 berikut.

Gambar 4.4 *Popup Menu Profile* Setelah Perbaikan

4.1.2. Pengembangan Sistem (*Construction*)

Pada tahap *Construction*, pembangunan aplikasi dilakukan secara bertahap berdasarkan desain yang telah disepakati sebelumnya. Pengembangan *frontend* menggunakan bahasa pemrograman *Flutter*, sedangkan pengelolaan sisi *server (backend)* menggunakan *framework Express.js*. Data disimpan menggunakan *MongoDB*, sementara untuk mendukung pemantauan kualitas air secara *real-time*, aplikasi diintegrasikan dengan *Firebase Real-time Database*. Berikut ini merupakan penjelasan lengkap mengenai tampilan antarmuka aplikasi yang telah dikembangkan, beserta pembahasan terkait fungsionalitas dari setiap fitur aplikasi.

1. Halaman *Splash Screen*



Gambar 4.5 Halaman *Splash Screen*

Gambar 4.5 merupakan halaman *Splash Screen* yang akan muncul pertama kali ketika aplikasi Sadewa SmartFarm dibuka. Halaman ini akan menentukan navigasi selanjutnya berdasarkan

sesi yang tersimpan pada aplikasi. Berikut ini merupakan kode halaman *Splash Screen* yang ditampilkan pada Kode 4.1.

Kode 4.1 Halaman *Splash Screen*

```
Future<void> _splashScreenLogic() async {
    await Future.Delayed(const Duration(seconds: 3));

    final prefs = await
SharedPreferences.getInstance();
    final token = prefs.getString('token');
    final role = prefs.getString('role');
    if (token != null && token.isNotEmpty &&
!JwtDecoder.isExpired(token)) {
        if (role == "Admin") {
            Navigator.of(context).push
Replacement(MaterialPageRoute(builder: (_) => const
BerandaAdmin()));
        } else {
            Navigator.of(context).push
Replacement(MaterialPageRoute(builder: (_) => const
BerandaUser()));
        }
    } else {
        await prefs.remove('token');
        await prefs.remove('role');
        Navigator.of(context).push
Replacement(MaterialPageRoute(builder: (_) => const
Login())));
    }
}
```

Kode 4.1 di atas merupakan bagian dari *method* *_splashScreenLogic()* dalam halaman *Splash Screen*, yang digunakan untuk memeriksa token dan peran pengguna yang tersimpan di *SharedPreferences*. Jika token valid dan belum kedaluwarsa, pengguna akan diarahkan ke halaman Beranda sesuai dengan peran (admin atau *user*). Jika token tidak ada atau sudah tidak valid, pengguna akan diarahkan ke halaman *Login*.

2. Halaman *Login*



Gambar 4.6 Halaman *Login*

Gambar 4.6 merupakan halaman *Login* yang digunakan untuk masuk ke aplikasi. Pada halaman ini pengguna harus memasukkan *username* dan *password* yang terdaftar pada sistem untuk masuk ke aplikasi. Ketika pengguna telah memasukkan *username* dan *password* lalu menekan tombol *Login*, aplikasi akan melakukan pengecekan apakah kredensial yang dimasukkan oleh pengguna valid atau tidak. Jika kredensial terdaftar pada sistem, maka aplikasi akan membawa pengguna ke halaman Beranda Admin atau *User* sesuai dengan *role* pengguna yang melakukan *login*. Pada halaman *Login*, pengguna juga dapat mengakses fitur Lupa *Password* jika mereka lupa atau tidak dapat mengingat *password* untuk *login*. Ketika pengguna menekan opsi Lupa *Password*, aplikasi akan mengarahkan pengguna ke halaman Lupa *Password*. Berikut ini merupakan kode halaman *Login* yang ditampilkan pada Kode 4.2.

Kode 4.2 Halaman *Login*

```

void _login() async {
    String username = usernameController.text.trim();
    String password = passwordController.text.trim();

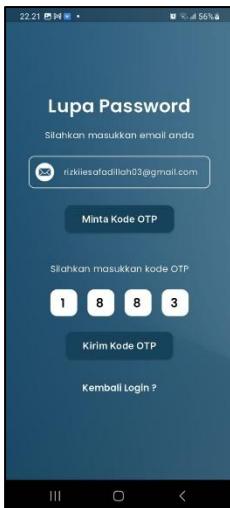
    bool success = await ApiService.login(username,
    password);
    if (!mounted) return;
    setState(() => isLoading = false);
    if (success) {
        SharedPreferences prefs = await
        SharedPreferences.getInstance();

        String? role = prefs.getString("role");
        _showDialog(true, "Login berhasil!",
    onComplete: () {
        if (role == "Admin") {
            Navigator.of(context).push Replacement(
                MaterialPageRoute(builder: (context) =>
        const BerandaAdmin()),
            );
        } else {
            Navigator.of(context).push Replacement(
                MaterialPageRoute(builder: (context) =>
        const BerandaUser()),
            );
        }
    });
    } else {
        _showDialog(false, "Login gagal. Periksa
    kembali username/password.");
    }
}
}

```

Kode 4.2 di atas merupakan implementasi dari *method* *_login()* pada halaman *Login*, yang digunakan untuk menangani proses *login* pengguna. Pertama, *method* ini memvalidasi apakah *input username* dan *password* tidak kosong. Jika valid, *method* melanjutkan proses *autentikasi* dengan memanggil fungsi *login()* dari *ApiService*. Setelah proses *login* berhasil, peran (*role*) pengguna diambil dari *SharedPreferences* untuk menentukan halaman yang dituju.

3. Halaman Lupa Password



Gambar 4.7 Halaman Lupa Password

Gambar 4.7 merupakan halaman Lupa Password yaitu halaman yang digunakan oleh pengguna untuk melakukan proses pemulihan akun jika lupa kata sandi. Pada halaman ini, pengguna diminta untuk memasukkan alamat *email*, kemudian menekan tombol Minta *OTP* (*One-Time Password*). Aplikasi kemudian akan mengirimkan kode *OTP* ke alamat *email* yang telah dimasukkan. Setelah kode *OTP* terkirim, pengguna dapat memasukkan kode *OTP* yang diterima melalui *email* ke kolom yang tersedia. Jika kode *OTP* yang dimasukkan sesuai, pengguna akan diarahkan ke halaman *Reset Password*. Berikut ini merupakan kode halaman Lupa Password yang ditampilkan pada Kode 4.3.

Kode 4.3 Halaman Lupa *Password*

```

void _requestOTP() async {
    String email = emailController.text.trim();
    bool success = await ApiService.sendOTP(email);
    if (success) {
        showDialog(true, "Kode OTP telah dikirim ke
email Anda. Silakan cek email Anda.");
    } else {
        showDialog(false, "Gagal mengirim OTP.
Pastikan email sudah terdaftar.");
    }
}

void _verifyOTP() async {
    String? response = await ApiService.verifyOTP(
        emailController.text, OTPController.text,
    );
    if (response == "expired") {
        showDialog(false, "Kode OTP telah kedaluwarsa!
Silakan minta kode baru.");
    } else if (response != null) {
        Navigator.of(context).push Replacement(
            MaterialPageRoute(builder: (context) => Reset
password(token: response)),
        );
    } else {
        showDialog(false, "Kode OTP yang Anda masukkan
salah!");
    }
}

```

Kode 4.3 di atas merupakan implementasi dari *method* *_requestOTP()* yang digunakan untuk memvalidasi *input email* dan mengirimkan *OTP* ke alamat *email* yang telah terdaftar menggunakan *ApiService.sendOTP()* dan *method* *_verifyOTP()* berfungsi untuk memverifikasi kode *OTP* yang dimasukkan oleh pengguna.

4. Halaman *Reset password*



Gambar 4.8 Halaman *Reset password*

Gambar 4.8 merupakan halaman *Reset password* yang dapat diakses pengguna untuk membuat *password* baru dengan memasukkan *password* yang diinginkan ke dalam kolom yang tersedia. Pengguna diminta untuk mengkonfirmasi *password* baru tersebut dengan memasukkannya kembali ke kolom konfirmasi *password*. Setelah pengguna menekan tombol simpan, pengguna akan diarahkan kembali ke halaman *Login* dan dapat menggunakan *password* tersebut untuk *login* ke aplikasi. Fitur ini dirancang untuk memastikan keamanan akun pengguna sekaligus memberikan kemudahan dalam proses pemulihan akses ke aplikasi. Berikut ini merupakan kode halaman *Reset password* yang ditampilkan pada Kode 4.4.

Kode 4.4 Halaman *Reset password*

```
void _resetPassword() async {
    String password = passwordController.text;
    String confirmPassword =
confirmPasswordController.text;

    bool success = await ApiService.reset
password(widget.token, password);
    if (success) {
        showDialog(true, "Password berhasil
diperbarui. Silakan login dengan password baru.",
onComplete: () {
            Navigator.of(context).pushReplacement(
                MaterialPageRoute(builder: (context) =>
const Login()),
            );
        });
    } else {
        showDialog(false, "Gagal memperbarui password.
Silakan coba lagi.");
    }
}
```

Kode 4.4 di atas merupakan implementasi *method _reset password()* pada halaman *Reset password*, yang bertugas memvalidasi *input password* dan *konfirmasi password* sebelum mengirimkan permintaan pembaruan kata sandi ke *server*. *Method* ini memeriksa validasi *password* yang baru. Jika semua validasi terpenuhi, *method* akan memanggil *ApiService.reset password()* dengan token dan *password* baru.

5. Halaman Beranda Admin



Gambar 4.9 Halaman Beranda Admin

Gambar 4.9 merupakan halaman Beranda Admin yang hanya dapat diakses oleh pengguna dengan *role* admin. Pada halaman ini, terdapat penunjuk waktu dan tanggal yang ditampilkan secara *real-time*, membantu pengguna untuk mengetahui waktu terkini saat mengakses sistem. Fitur ini terletak di bagian atas halaman, bersama dengan *widget profile* untuk mengakses menu *profile*. Selain itu, halaman ini menampilkan daftar kolam yang dapat dimonitoring oleh pengguna. Pada halaman Beranda Admin, selain dapat memonitoring kolam, admin juga dapat menambahkan kolam baru yang akan dimonitoring melalui tombol Tambah Kolam. Pada *tile* kolam juga terdapat ikon titik tiga di pojok kanan atas untuk menampilkan *widget* menu kolam. Pada bagian bawah halaman, terdapat menu navigasi ke halaman

Beranda dan Manajemen *User*. Berikut ini merupakan kode halaman Beranda Admin yang ditampilkan pada Kode 4.5.

Kode 4.5 Halaman Beranda Admin

```
Future<void> fetchPonds() async {
    final data = await ApiService.getKolam();
    setState(() {
        pondList = data;
    });
}
```

Kode 4.5 di atas merupakan implementasi *method* *fetchPonds()* yang digunakan untuk mengambil data daftar kolam melalui *ApiService.getKolam()*.

6. Halaman Beranda *User*



Gambar 4.10 Halaman Beranda *User*

Gambar 4.10 merupakan halaman Beranda *User* yang hanya dapat diakses oleh pengguna dengan *role user*. Tampilan dan fitur yang terdapat pada halaman ini hampir sama dengan halaman

Beranda Admin. Namun, pada halaman ini tidak ada tombol Tambah Kolam, *Popup* menu kolam, dan menu navigasi di bagian bawah halaman. Berikut ini merupakan kode halaman Beranda *User* yang ditampilkan pada Kode 4.6.

Kode 4.6 Halaman Beranda *User*

```
Future<void> fetchPonds() async {
    final data = await ApiService.getKolam();
    setState(() {pondList = data;});
}
```

Kode 4.6 di atas merupakan implementasi dari *method* *fetchPonds()* yang digunakan untuk mengambil data kolam dari *server* melalui pemanggilan *ApiService.getKolam()*.

7. *Popup* Menu Kolam



Gambar 4.11 *Popup* Menu Kolam

Gambar 4.11 menampilkan *Popup* Menu Kolam yang muncul saat admin menekan ikon titik tiga di pojok kanan atas pada *tile* kolam yang dipilih dari daftar kolam. Menu ini menyediakan dua

opsi utama bagi Admin, yaitu Hapus dan Edit kolam, yang memungkinkan Admin untuk mengelola data kolam secara langsung dari halaman Beranda Admin. Berikut ini merupakan kode *Popup* Menu Kolam yang ditampilkan pada Kode 4.7.

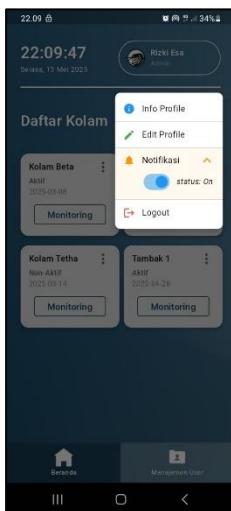
Kode 4.7 *Popup* Menu Kolam

```
void _EditPond(BuildContext context, Map<String,
dynamic> pond) async {
    final updatedPond = await Navigator.push(
        context,
        MaterialPageRoute(
            builder: (context) => EditKolam(
                id: pond["_id"],
                pondId: pond["idPond"],
                pondName: pond["namePond"],
                status: pond["statusPond"],
            ),
        ),
    );
    if (updatedPond != null) {
        fetchPonds();
    }
}

void _DeletePond(String id, String pondName) {
    CustomDialogButton.show(
        context: context,
        title: "Konfirmasi Hapus",
        message: "Apakah Anda yakin ingin menghapus
kolam \"\$pondName\"? Tindakan ini tidak dapat
dibatalkan.",
        confirmText: "Hapus",
        cancelText: "Batal",
        isWarning: true,
        onConfirm: () async {
            bool deleted = await
        ApiService.deleteKolam(id);
            if (deleted) {
                setState(() {
                    pondList.removeWhere((pond) =>
                pond["_id"] == id);
            });
        }
    );
}
```

Kode 4.7 di atas merupakan implementasi dari dua *method*, yaitu *_EditPond()* yang berfungsi untuk menavigasi pengguna ke halaman Edit Kolam dengan membawa data kolam yang ingin diedit dan *_DeletePond()* yang digunakan untuk menghapus kolam berdasarkan *id* dengan menampilkan dialog konfirmasi. Jika pengguna menyetujui data kolam akan dihapus melalui *ApiService.deleteKolam()*.

8. Popup Menu Profile



Gambar 4.12 Popup Menu Profile

Gambar 4.12 menampilkan *Popup Menu Profile* yang muncul saat pengguna menekan *widget profile* pada pojok kanan atas halaman Beranda Admin atau *User*. *Popup Menu Profile* menyediakan empat opsi, yaitu *Info Profile*, *Edit Profile*, *Notifikasi*, dan *Logout*. Untuk fitur Notifikasi sendiri terdapat dropdown yang berisi tombol *switch* untuk menghidupkan atau mematikan

status/*popup* notifikasi, Berikut ini merupakan kode *Popup Menu Profile* yang ditampilkan pada Kode 4.8.

Kode 4.8 *Popup Menu Profile*

```
void _infoProfile() {
    MyApp.navigatorKey.currentState?.push (
        MaterialPageRoute(builder: (context) => const
Profile()),
    );
}
void _editProfile() {
    MyApp.navigatorKey.currentState?.push (
        MaterialPageRoute(builder: (context) => const
EditProfile()),
    );
}
Future<void> _checkNotificationStatus() async {
    final userId = await _getCurrentUserId();
    if (userId == null) return;
    final prefs = await
SharedPreferences.getInstance();
    final status =
prefs.getBool('notifications_enabled_$userId') ??
true;
    setState(() {_notificationsEnabled = status;
    });
}
void _logout() async {
    CustomDialogButton.show(
        context: MyApp.navigatorKey.currentContext!,
        title: "Konfirmasi Logout",
        message: "Apakah Anda yakin ingin logout?",
        confirmText: "Ya",
        onConfirm: () async {
            final prefs = await
SharedPreferences.getInstance();
            await prefs.remove('notifications_enabled');
            await ApiService.logout();
            MyApp.navigatorKey.currentState?.push
AndRemoveUntil(
                MaterialPageRoute(builder: (context) =>
const Login(),
                    (route) => false,);
        },
        cancelText: "Batal",isWarning: true,
    );
}
```

Kode 4.8 di atas merupakan implementasi beberapa *method* yang menangani fitur *profile* pengguna dan pengaturan notifikasi. *Method* *_InfoProfile()* dan *_EditProfile()* berfungsi untuk menavigasi ke halaman *Profile* dan *Edit Profile*. *Method* *_checkNotificationStatus()* digunakan untuk mengecek dan mengatur status notifikasi berdasarkan *id* pengguna. Terakhir, *method* *_logout()* untuk keluar dari aplikasi dan membersihkan data sesi.

9. Halaman Tambah Kolam



Gambar 4.13 Halaman Tambah Kolam

Gambar 4.13 merupakan halaman Tambah Kolam yang digunakan admin untuk menambahkan kolam baru yang akan di monitoring dengan mengisi *input id* Kolam dan Nama Kolam. *Id* kolam merupakan identitas unik yang membedakan satu kolam dengan kolam lainnya. Berikut ini merupakan kode halaman Tambah Kolam yang ditampilkan pada Kode 4.9.

Kode 4.9 Halaman Tambah Kolam

```
Future<void> _AddPond() async {
    String idPond = _idController.text.trim();
    String namePond = _nameController.text.trim();
    setState(() => _isLoading = true);
    bool success = await ApiService.addKolam(idPond,
namePond);
    setState(() => _isLoading = false);
    _showDialog(
        success, success ? "Kolam berhasil ditambahkan"
: "Gagal menambahkan kolam. Coba lagi!",
        onComplete: () {if (success) {widget
.onKolamAdded(); Navigator.pop(context);}
},
    );
}
```

Kode 4.9 di atas merupakan implementasi dari *method* *_AddPond()* yang digunakan untuk menambahkan data kolam baru. *Method* ini akan memproses penambahan kolam melalui *ApiService.addKolam()*.

10. Halaman Edit Kolam



Gambar 4.14 Halaman Edit Kolam

Gambar 4.14 merupakan halaman Edit Kolam yang digunakan admin untuk memperbarui nama kolam dan status kolam. Halaman ini menyediakan dua *input*, yaitu Nama Kolam dan Status Kolam. Admin dapat mengubah nama kolam sesuai kebutuhan serta mengatur status kolam menjadi Aktif atau Non-Aktif, tergantung pada kondisi kolam tersebut. Berikut inimerupakan kode halaman Edit Kolam yang ditampilkan pada Kode 4.10.

Kode 4.10 Halaman Edit Kolam

```
Future<void> _EditPond() async {
    String newName = nameController.text.trim();
    String newStatus = selectedStatus ?? "Aktif";

    setState(() => _isLoading = true);

    var updatedKolam = await
    ApiService.editKolam(widget.id, widget.pondId,
    newName, newStatus);
    if (!mounted) return;
    setState(() => _isLoading = false);

    if (updatedKolam != null) {
        _showDialog(true, "Kolam berhasil diperbarui!",
    onComplete: () {
        Navigator.pop(context, updatedKolam);
    });
    } else {
        _showDialog(false, "Gagal memperbarui kolam.
    Coba lagi.");
    }
}
```

Kode 4.10 di atas merupakan implementasi dari *method* *_EditPond()* yang digunakan untuk memperbarui data kolam. *Method* ini melakukan validasi nama kolam dan status. Jika valid, proses pembaruan dikirim ke *server* melalui *ApiService.editKolam()*.

11. Halaman *Profile*



Gambar 4.15 Halaman *Profile*

Gambar 4.15 merupakan halaman *Profile* yang dapat diakses pengguna untuk melihat data informasi pengguna seperti *Username*, *Email*, *Role* (Admin atau *User*), dan *Tanggal Daftar*. Informasi ini memberikan gambaran lengkap mengenai akun pengguna yang sedang *login*. Berikut ini merupakan kode halaman *Profile* yang ditampilkan pada Kode 4.11.

Kode 4.11 Halaman *Profile*

```
Future<void> fetchData() async {
    final data = await ApiService.getProfile();
    setState(() { userData = data; });
}
```

Kode 4.11 di atas merupakan implementasi dari *method* *fetchUserData()* yang digunakan untuk mengambil data pengguna melalui *ApiService.getProfile()*.

12. Halaman Edit *Profile*



Gambar 4.16 Halaman Edit *Profile*

Gambar 4.16 merupakan halaman Edit *Profile* yang dapat diakses pengguna untuk memperbarui data pribadi mereka, seperti *Username* dan *Email*, sementara informasi lain seperti *Role*, *password* dan Tanggal Daftar tidak dapat diedit pada halaman ini untuk menjaga integritas dan keamanan sistem. Berikut ini merupakan kode halaman Edit *Profile* yang ditampilkan pada Kode 4.12.

Kode 4.12 Halaman Edit *Profile*

```
Future<void> _EditProfile() async {
    String username =
    _usernameController.text.trim(); String email =
    _emailController.text.trim().toLowerCase();
    setState(() => _isLoading = true);
    bool success = await
    ApiService.editProfile(username, email);
    if (!mounted) return;
    if (success) {
        showDialog(true, "Profile berhasil
diperbarui!", onComplete: () {
        Navigator.of(context).push Replacement(
```

```

        MaterialPageRoute(builder: (context) =>
const Profile(),
);
}
} else {
_showDialog(false, "Gagal memperbarui profile.
Coba lagi.");
}
}
}

```

Kode 4.12 di atas merupakan implementasi dari *method _EditProfile()* yang digunakan untuk memperbarui informasi *profile* pengguna berupa *username* dan *email*. *Method* ini akan memvalidasi *input username* dan *email*. Jika validasi berhasil, maka *profile* pengguna akan diperbarui melalui *ApiService.editProfile()*.

13. Halaman Manajemen *User*



Gambar 4.17 Halaman Manajemen *User*

Gambar 4.17 merupakan Halaman Manajemen *User* yang hanya dapat diakses oleh pengguna dengan *role* Admin. Halaman

Manajemen *User* memungkinkan Admin untuk mengelola data pengguna yang terdaftar dalam aplikasi. Informasi pengguna yang ditampilkan dalam *card user* meliputi *username*, *email*, *role*, dan tanggal dibuat. Fitur pada halaman ini yaitu, Tambah *User*, Edit *User* dan Hapus *User*. Berikut ini merupakan kode halaman Manajemen *User* yang ditampilkan pada Kode 4.13.

Kode 4.13 Halaman Manajemen *User*

```
Future<void> _fetchUsers() async {
  List<Map<String, dynamic>> fetchedUsers = await
  ApiService.getUsers();
  setState(() { _users = fetchedUsers;});}
```

Kode 4.13 di atas merupakan implementasi dari *method* *_fetchUsers()* yang berfungsi untuk mengambil data seluruh pengguna dari *server* menggunakan *ApiService.getUsers()*.

14. Popup Menu *User*



Gambar 4.18 Popup Menu *User*

Gambar 4.18 menampilkan *Popup Menu User* yang muncul saat admin menekan ikon titik tiga di pojok kanan atas *card user*. *Popup* ini memiliki 2 menu yaitu Edit dan Hapus *User*. Menu Edit digunakan untuk memperbarui informasi akun yang sudah ada. Sementara itu, menu Hapus memungkinkan admin menghapus akun yang tidak lagi aktif atau tidak diperlukan. Berikut ini merupakan kode *Popup Menu User* yang ditampilkan pada Kode 4.14.

Kode 4.14 *Popup Menu User*

```
void _editUserPopup(Map<String, dynamic> user) {
    _overlayEntry?.remove();
    _overlayEntry = null;

    Navigator.of(context).push (MaterialPageRoute(
        builder: (_) => EditUser(
            userId: user['_id'],
            username: user['username'],
            email: user['email'],
            role: user['role'],
            onUpdateSuccess: _fetchUsers,
        ),
    )));
}

void _deleteUserPopup(Map<String, dynamic> user) {
    CustomDialogButton.show(
        context: context,
        title: "Konfirmasi Hapus",
        message: "Apakah Anda yakin ingin menghapus pengguna ini?",
        confirmText: "Hapus",
        cancelText: "Batal",
        isWarning: true,
        onConfirm: () async {
            bool success = await
                ApiService.deleteUser(user['_id']);
            if (success) _fetchUsers();
            _showDialog(success, success ? "Pengguna berhasil dihapus" : "Gagal menghapus pengguna");
        },
    );
}
```

Kode 4.14 di atas merupakan implementasi *method _editUserPopup()* yang digunakan untuk membuka halaman edit pengguna dengan data pengguna yang dipilih dan *method _deleteUserPopup()* yang berfungsi untuk menghapus pengguna tersebut melalui *ApiService.deleteUser()* jika disetujui.

15. Halaman Tambah *User*



Gambar 4.19 Halaman Tambah *User*

Gambar 4.19 merupakan halaman Tambah *User* yang hanya bisa diakses oleh Admin untuk melakukan pendaftaran akun baru dengan mengisi beberapa *input*, yaitu *Username*, *Email*, *Role* (Admin atau *User*), *Password*, dan *Confirm Password*. Melalui *input Role*, admin dapat menentukan peran (*role*) pengguna secara langsung saat proses pendaftaran. Berikut ini merupakan kode halaman Tambah *User* yang ditampilkan pada Kode 4.15.

Kode 4.15 Halaman Tambah *User*

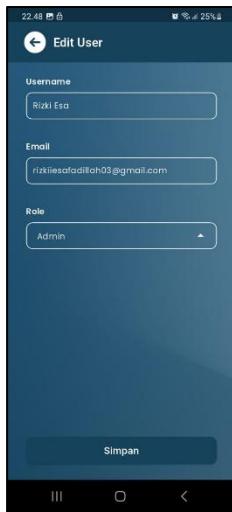
```
Future<void> _AddUser() async {
    String username = usernameController.text.trim();
    String email = emailController.text.trim();
    String password = passwordController.text;
    String confirmPassword =
confirmPasswordController.text;

    bool success = await ApiService.addUser(username,
email, password, selectedRole!);

    setState(() => isLoading = false);
    showDialog(
        success,
        success ? "User berhasil ditambahkan" : "Gagal
menambahkan user",
        onComplete: () {
            if (success) {
                widget.onUserADded();
                Navigator.pop(context);
            }
        },
    );
}
```

Kode 4.15 di atas merupakan implementasi *method* *_AddUser()* yang berfungsi untuk menambahkan pengguna baru. *Method* tersebut terdapat validasi lengkap pada *input username, email, password*, dan konfirmasi *password*. Jika semua validasi berhasil, data akan dikirim ke *ApiService.addUser()* untuk menambahkan pengguna.

16. Halaman Edit *User*



Gambar 4.20 Halaman Edit *User*

Gambar 4.20 merupakan halaman Edit *User* yang hanya bisa diakses Admin untuk memperbarui data pengguna aplikasi. Pada halaman ini, terdapat *input* seperti *Username*, *Email*, dan *Role*. Admin dapat mengubah *username* dan *email* pengguna, serta menyesuaikan *role* (Admin atau *User*) sesuai dengan kebutuhan. Berikut ini merupakan kode halaman Edit *User* yang ditampilkan pada Kode 4.16.

Kode 4. 16 Halaman Edit *User*

```
Future<void> _EditUser() async {
    String username =
    _usernameController.text.trim();
    String email = _emailController.text.trim();
    String role = _selectedRole ?? widget .role;

    setState(() => _isLoading = true);
    bool success = await ApiService.editUser(widget
        .userId, username, email, role);
    if (!mounted) return;
```

```

    setState(() => _isLoading = false);
    if (success) {
        showDialog(
            true,
            "Data pengguna berhasil diperbarui!",
            onComplete: () {
                widget.onUpdateSuccess();
                Navigator.pop(context);
            },
        );
    } else {
        showDialog(false, "Gagal memperbarui data
pengguna. Coba lagi.");
    }
}

```

Kode 4.16 di atas merupakan implementasi *method* *_EditUser()* yang digunakan untuk memperbarui data pengguna. Fungsi ini melakukan validasi *input username* dan *email*. Jika validasi berhasil, data pengguna akan dikirim ke *ApiService.editUser()* untuk diperbarui.

17. Halaman Monitoring



Gambar 4.21 Halaman Monitoring

Gambar 4.21 merupakan halaman Monitoring yang menampilkan informasi terkini dari setiap sensor yang digunakan untuk memantau kondisi kolam budidaya udang. Pada halaman ini, terdapat empat parameter utama yang dipantau, yaitu sensor suhu dengan satuan *celcius*, sensor kekeruhan/*turbidity* dengan satuan *NTU*, sensor pH, dan sensor salinitas dengan satuan *PPT*. Berikut ini merupakan kode halaman Manajemen Monitoring yang ditampilkan pada Kode 4.17.

Kode 4.17 Halaman Monitoring

```
void fetchSensorData() async {
    try {
        final data = await
        ApiService.getMonitoringData(widget .pondId);
        if (mounted) {
            setState(() {
                _sensorStore.updateSensorHistory(widget
                .pondId, "temperature", data["temperature"]);
                _sensorStore.updateSensorHistory(widget
                .pondId, "ph", data["ph"]);
                _sensorStore.updateSensorHistory(widget
                .pondId, "salinity", data["salinity"]);
                _sensorStore.updateSensorHistory(widget
                .pondId, "turbidity", data["turbidity"]);

                _sensorStore.setSensorData(widget .pondId,
                data);
                isLoading = false;
            });
        }
    } catch (e) {
        print("Error: $e");
        if (mounted) {
            setState(() => isLoading = false);
        }
    }
}
```

Kode 4.17 di atas merupakan implementasi *method* *fetchSensorData()* yang mengambil data sensor monitoring kolam

dari `ApiService.getMonitoringData()`, lalu memperbarui `state` aplikasi dengan data tersebut. Data suhu, pH, salinitas, dan kekeruhan disimpan ke dalam `store` sensor untuk histori, dan data sensor terkini juga disimpan.

18. Halaman Pengaturan Batasan Sensor



Gambar 4.22 Halaman Pengaturan Batasan Sensor

Gambar 4.22 merupakan halaman Pengaturan Sensor, dimana pada halaman ini akan menampilkan nilai sensor saat ini, serta batasan nilai sensor tertinggi dan terendah. Pengguna dapat mengatur batasan nilai tertinggi dan nilai terendah yang diinginkan ataupun sesuai dengan yang direkomendasikan melalui kolom *inputan* yang telah disediakan. Pada bagian bawah kolom pengaturan batasan parameter sensor, terdapat fitur Informasi Perangkat Sensor yang dapat diklik untuk menampilkan informasi

tentang perangkat sensor. Berikut ini merupakan kode halaman Pengaturan Batasan Sensor yang ditampilkan pada Kode 4.18.

Kode 4.18 Halaman Pengaturan Batasan Sensor

```
Future<void> saveThresholdValues() async {
    try {
        final sensorType = widget
    .sensorType.toLowerCase();
        final highValueFormatted =
tempHighValue?.toStringAsFixed(1);
        final lowValueFormatted =
tempLowValue?.toStringAsFixed(1);
        await ApiService.updateThresholds(widget
    .pondId, {
        sensorType: {
            "high": double.parse(highValueFormatted!),
            "low": double.parse(lowValueFormatted!),
        }
    });
        setState(() {
            highestValue = tempHighValue;
            lowestValue = tempLowValue;
        });
        CustomDialog.show(
            context: context,
            isSuccess: true,
            message: "Batasan sensor
${getReadableSensorName(widget .sensorType)} berhasil diperbarui.",
        );
    } catch (e) {
        CustomDialog.show(
            context: context,
            isSuccess: false,
            message: "Gagal menyimpan data:
${e.toString()}",
        );
    }
}
```

Kode 4.18 di atas ini merupakan implementasi dari fungsi *saveThresholdValues()* yang digunakan menyimpan perubahan ambang batas sensor tertinggi dan terendah ke *firebase* melalui *ApiService.updateThresholds()*.

19. Halaman Informasi Sensor



Gambar 4.23 Halaman Informasi Sensor

Gambar 4.23 merupakan halaman Informasi Sensor yang berisi deskripsi tentang sensor yang digunakan, spesifikasi perangkat, serta rentang suhu optimal sensor untuk budidaya udang. Dengan adanya fitur ini, pengguna dapat memahami lebih dalam tentang perangkat sensor yang digunakan dalam pemantauan kolam budidaya udang. Berikut ini merupakan kode halaman Informasi Sensor yang ditampilkan pada Kode 4.19.

Kode 4.19 Halaman Informasi Sensor

```
KolomInformasiSensor(
    sensorTitle: sensorData["title"] ?? "Sensor Tidak
Diketahui",
    imagePath: sensorData["image"] ?? 
"assets/images/default-sensor.jpg",
    description: sensorData["description"] ?? 
"Informasi sensor tidak tersedia.",
    specifications:
List<String>.from(sensorData["specifications"] ?? 
[]),
```

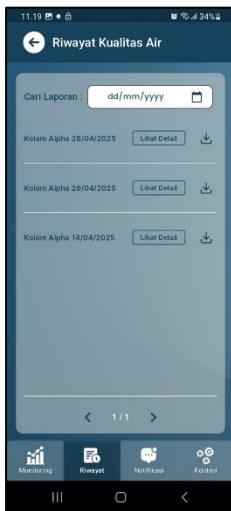
```

    optimalRange: sensorData["optimalRange"] ??
    "Tidak ada data rentang optimal.",
    rangeTitle: sensorData["rangeTitle"] ?? "Rentang
Optimal",
),

```

Kode 4.19 di atas ini merupakan pemanggilan *widget KolomInformasiSensor* yang digunakan untuk menampilkan informasi detail sebuah sensor, termasuk judul, gambar, deskripsi, spesifikasi, dan rentang optimal penggunaannya.

20. Halaman Riwayat Data Monitoring



Gambar 4.24 Halaman Riwayat Data Monitoring

Gambar 4.24 merupakan halaman Riwayat Data Monitoring yang menampilkan Daftar data monitoring tambak udang setiap harinya. Pada halaman ini pengguna dapat mencari data riwayat kualitas air tambak yang diinginkan baik secara manual atau melalui fitur kolom pencarian data berdasarkan tanggal. Setiap data dapat dilihat dengan menekan tombol lihat detail dan

mengunduh data riwayat tambak dalam format *excel* dengan menekan ikon unduh. Pada bagian bawah daftar Riwayat terdapat tombol *paginasi* yang dapat membantu pengguna dalam memilih riwayat lainnya. Berikut ini merupakan kode halaman Riwayat Data Monitoring yang ditampilkan pada Kode 4.20.

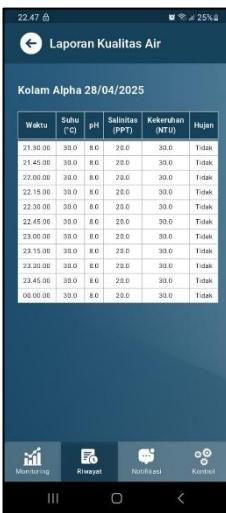
Kode 4.20 Halaman Riwayat Data Monitoring

```
Future<void> _fetchHistory() async {
    setState(() {
        isLoading = true;
    });
    List<dynamic>? response = await
        ApiService.getHistoryByPond(widget .pondId);
    setState(() {
        if (response != null && response.isNotEmpty) {
            historyData = response.map((item) {
                DateTime parsedDate =
                    DateTime.parse(item["date"]);
                String formattedDate =
                    "${parsedDate.day.toString().padLeft(2, '0')}/"
                    "${parsedDate.month.toString().padLeft(2, '0')}/"
                    "${parsedDate.year}";
                return {
                    "id": item["_id"],
                    "date": formattedDate,
                    "data": item["data"],
                };
            }).toList();
            filteredData = List.from(historyData);
        }
        isLoading = false;
    });
}
```

Kode 4.20 di atas ini merupakan implementasi dari beberapa *method* yang digunakan untuk menampilkan dan mencari riwayat berdasarkan tanggal pada halaman histori kolam. *Method* *_fetchHistory()* mengambil data histori dari *ApiService.getHistoryByPond()* berdasarkan *id* kolam, memformat

tanggalnya, lalu menyimpannya ke dalam *state historyData* dan *filteredData*. Method *_filterData()* berfungsi untuk menyaring data histori berdasarkan tanggal yang dipilih.

21. Halaman Laporan Data Monitoring



Gambar 4.25 Halaman Laporan Data Monitoring

Gambar 4.25 merupakan halaman Laporan Data Monitoring yang menampilkan data kualitas air tambak yang diperbarui setiap 15 menit. Kolom data laporan ini mencakup informasi penting, seperti waktu pencatatan, suhu air, pH, salinitas, kekeruhan, dan status hujan yang dapat membantu pengguna dalam menelusuri dan menganalisis perubahan kondisi air dari waktu ke waktu, sehingga dapat dijadikan acuan dalam pengambilan keputusan terkait pengelolaan tambak secara berkelanjutan. Berikut ini merupakan kode halaman Laporan Data Monitoring yang ditampilkan pada Kode 4.21.

Kode 4.21 Halaman Laporan Data Monitoring

```
Future<List<Map<String, dynamic>>>
_fetchLaporanData() async {
    try {
        if (widget .id.isEmpty) return [];
        final response = await
        ApiService.getHistoryById(widget .id);
        if (response != null && response["data"] is
List) {
            return (response["data"] as List)
                .map((item) => {
                    "waktu": item["time"] ?? "-",
                    "suhu": (item["temperature"] as
num?)?.toStringAsFixed(1) ?? "0.0",
                    "ph": (item["ph"] as
num?)?.toStringAsFixed(1) ?? "0.0",
                    "salinitas": (item["salinity"] as
num?)?.toStringAsFixed(1) ?? "0.0",
                    "kekeruhan": (item["turbidity"] as
num?)?.toStringAsFixed(1) ?? "0.0",
                    "hujan": item["rain_status"] ?? false,
                })
                .toList();
        }
    } catch (e) {
        debugPrint("Error fetching data: $e");
    }
    return [];
}
```

Kode 4.21 di atas ini merupakan implementasi fungsi *_fetchLaporanData()* yang digunakan untuk mengambil data histori sensor berdasarkan *id* tertentu. Data yang diterima dari *ApiService.getHistoryById()* akan diolah menjadi daftar berisi waktu, suhu, pH, salinitas, kekeruhan, dan status hujan.

22. Halaman *Notifikasi*



Gambar 4.26 Halaman *Notifikasi*

Gambar 4.26 merupakan halaman *Notifikasi* yang menampilkan daftar peringatan dan pemberitahuan terkait kondisi tambak udang. *Notifikasi* ini mencakup berbagai informasi penting, seperti status jumlah pakan, kualitas air, perubahan data batasan parameter sensor, jadwal dan jumlah pakan, serta delay aerator. Di bagian kiri halaman, menunjukkan daftar notifikasi yang masing-masing dilengkapi dengan waktu terjadinya *notifikasi*. Sementara itu, di bagian kanan halaman, pengguna dapat melihat detail *notifikasi* yang dipilih. Selain itu, di bagian bawah daftar *notifikasi* terdapat tombol *paginasi* yang dapat membantu pengguna dalam untuk melihat pemberitahuan lainnya. Berikut ini merupakan kode halaman *Notifikasi* yang ditampilkan pada Kode 4.22.

Kode 4.22 Halaman *Notifikasi*

```
Future<void> _fetchNotifications() async {
    final notifications = await
    ApiService.getNotificationsByPondId(widget
    .pondId);
    if (notifications != null) {
        setState(() {
            notifikasiList =
            notifications.map<Map<String, dynamic>>((notif) {
                String jenis =
                _getNotificationType(notif["type"]);
                Color warna = jenis == "Peringatan" ?
                Colors.red : Colors.amber;
                return {
                    "id": notif["_id"],
                    "jenis": jenis,
                    "judul": notif["title"] ?? "Notifikasi",
                    "waktu": formatTime(notif["time"]),
                    "message": notif["message"] ?? "Tidak ada
pesan",
                    "warna": warna,
                    "status": notif["status"] ?? "unread",
                };
            }).toList();
        });
    }
}
```

Kode 4.22 berikut merupakan implementasi *method* *_fetchNotifications()* dan *_fetchNotificationDetail()* yang berfungsi untuk mengambil daftar notifikasi berdasarkan *id* kolam dan menampilkan detail notifikasi terpilih. *Method* *_fetchNotifications()* memanggil *ApiService.getNotificationsByPondId()* untuk mendapatkan data notifikasi. Sedangkan *_fetchNotificationDetail()* mengambil detail notifikasi tertentu, memperbarui status notifikasi menjadi sudah dibaca melalui API, dan memperbarui tampilan *UI* secara dinamis.

23. Halaman Kontrol Pakan dan *Aerator*



Gambar 4.27 Halaman Kontrol Pakan dan *Aerator*

Gambar 4.27 merupakan halaman Kontrol Pakan dan *Aerator*, yang dirancang untuk membantu pengguna dalam mengatur jadwal pemberian pakan udang serta pengoperasian *aerator* di tambak secara otomatis. Pada bagian Kontrol Pakan, pengguna dapat mengatur jadwal pemberian pakan. Selain itu, tersedia fitur untuk mengaktifkan atau menonaktifkan sistem pemberian pakan melalui tombol ON/OFF Pakan, serta menentukan jumlah pakan yang diberikan dalam satu kali pemberian. Setelah melakukan perubahan, pengguna dapat menyimpan pengaturan dengan menekan tombol Simpan. Sementara itu, pada bagian Kontrol *Aerator*, pengguna dapat mengelola pengoperasian *aerator* yang berfungsi untuk menjaga kadar oksigen dalam air. Fitur yang tersedia mencakup tombol ON/OFF *Aerator* untuk mengaktifkan atau menonaktifkan *aerator*, serta pengaturan *interval* waktu

pengoperasian setelah pemberian pakan, misalnya selama 15 menit. Setelah menyesuaikan pengaturan, pengguna dapat menyimpannya dengan menekan tombol Simpan. Berikut ini merupakan kode halaman Kontrol Pakan dan *Aerator* yang ditampilkan pada Kode 4.23.

Kode 4.23 Halaman Kontrol Pakan dan *Aerator*

```
Future<void> _updateAeratorConfig() async {
    setState(() => isSaving = true);
    try {
        await ApiService.updateAerator(
            widget.pondId,
            {
                "aerator_Delay": aeratorDelay?.toInt(),
            },
        );
        _showDialog(true, "Waktu Delay aerator berhasil
diperbarui");
    } catch (e) {
        print("Error saat memperbarui Delay aerator:
\$e");
        _showDialog(false, "Terjadi kesalahan saat
menyimpan Delay aerator");
    }
    setState(() => isSaving = false);
}

Future<void> _updateFeederConfig() async {
    setState(() => isSaving = true);
    try {
        final formattedSchedule =
        _feedingSchedule.map((time) {
            return time != null
                ? "${time.hour.toString().padLeft(2,
'0')}:${time.minute.toString().padLeft(2, '0')}"
                : null;
        }).toList();
        final Map<String, dynamic> payload = {
            "amount": feedAmount.toInt(),
            "schedule": formattedSchedule,
        };
        await ApiService.updateFeeding(widget.pondId,
payload);
        _showDialog(true, "Konfigurasi berhasil
disimpan");
    } catch (e) {
```

```

        print("Error saving config: $e");
        showDialog(false, "Gagal menyimpan
konfigurasi: ${e.toString()}");
    } finally {
        setState(() => isSaving = false);
    }
}

```

Kode 4.23 di atas ini merupakan implementasi dari beberapa *method* untuk memperbarui konfigurasi *aerator* dan *feeder* pada kolam. *Method updateAeratorConfig()* dan *_updateFeederConfig()* mengirim data konfigurasi yang sudah diubah ke *server*.

4.1.3. Implementasi (*Cutover*)

Setelah tahapan pengembangan sistem selesai, selanjutnya akan masuk ke dalam tahap implementasi dan pengujian. Sebelum di implementasikan, sistem akan diuji menggunakan 2 metode pengujian, yaitu *Black Box Testing* dan *System Usability Scale (SUS)* untuk menguji apakah semua fungsionalitas dari aplikasi serta menguji kepuasan Stakeholder Sadewa Farm terhadap aplikasi yang sudah dikembangkan.

4.1.3.1. Hasil Pengujian *Black Box Testing*

Pengujian *Black Box Testing* dilakukan untuk memverifikasi bahwa setiap fungsi aplikasi bekerja sesuai dengan spesifikasi yang telah ditentukan. Pengujian ini di fokuskan pada *input* dan *output* aplikasi tanpa memperhatikan struktur internal atau kode program. Hasil dari pengujian ini akan dicatat dalam tabel yang menunjukkan setiap fitur yang diuji, skenario pengujian, hasil yang diharapkan dan

status pengujian (berhasil/gagal). Hasil pengujian Black box pada aplikasi ini adalah sebagai berikut:

Tabel 4.3 Hasil Pengujian *Black Box Testing*

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
SK-01-1	<i>Login</i>	Pengguna memasukkan <i>email</i> dan <i>password</i> yang benar.	Aplikasi menampilkan pesan “ <i>Login berhasil!</i> ” dan pengguna diarahkan ke halaman Beranda jika memiliki <i>role Admin</i> dan ke halaman Beranda <i>User</i> jika memiliki <i>role User</i> .	Berhasil
SK-01-2		Pengguna memasukkan <i>username</i> dan/atau <i>password</i> yang salah.	Aplikasi menampilkan pesan <i>error</i> “ <i>Login gagal. Periksa kembali username/password.</i> ”	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
SK-01-3		Pengguna memasukkan <i>username</i> dan <i>password</i> dengan salah satu <i>field</i> kosong.	Aplikasi menampilkan pesan <i>error</i> “ <i>Username</i> dan <i>password</i> tidak boleh kosong.”	Berhasil
SK-02-1	Lupa <i>Password</i>	Pengguna menekan tombol teks “Lupa <i>Password</i> ” pada halaman <i>login</i> .	Aplikasi menampilkan halaman Lupa <i>Password</i> .	Berhasil
SK-02-2		Pengguna memasukkan <i>email</i> yang terdaftar untuk menerima kode <i>OTP</i> .	Aplikasi menampilkan pesan “Kode OTP telah dikirim ke email Anda. Silakan cek email Anda.” dan pengguna	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
			menerima kode <i>OTP</i> .	
SK-02-3		Pengguna memasukkan <i>email</i> yang tidak terdaftar.	Aplikasi menampilkan pesan <i>error</i> “Gagal mengirim <i>OTP</i> . Pastikan <i>email</i> sudah terdaftar.”	Berhasil
SK-02-4		Pengguna memasukkan kode <i>OTP</i> yang benar.	Pengguna berhasil masuk ke halaman <i>Reset Password</i> .	Berhasil
SK-02-5		Pengguna memasukkan kode <i>OTP</i> yang salah atau kadaluarsa.	Pengguna gagal masuk ke halaman <i>Reset Password</i> dan Aplikasi menampilkan pesan <i>error</i> .	Berhasil
SK-02-6		Pengguna tidak memasukkan <i>email</i> atau kode <i>OTP</i>	Aplikasi menampilkan pesan <i>error</i> .	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
		dan langsung klik tombol.		
SK-03-1	<i>Reset Password</i>	Pengguna memasukkan <i>password</i> baru dan <i>confirm password</i> yang valid.	Aplikasi menampilkan pesan “ <i>Password</i> berhasil diperbarui. Silakan <i>login</i> dengan <i>password</i> baru.” dan pengguna kembali ke halaman <i>login</i> .	Berhasil
SK-03-2		Pengguna memasukkan <i>password</i> baru dan/atau <i>confirm password</i> dengan data yang tidak valid atau kosong.	Aplikasi menampilkan pesan <i>error</i> .	Berhasil
SK-04-1	Halaman Beranda	Pengguna <i>login</i>	Aplikasi menampilkan	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
SK-04-2		menggunakan akun Admin.	halaman beranda dengan fitur-fitur dan menu yang lengkap, termasuk Manajemen <i>User</i> dan Manajemen Kolam.	
		Pengguna <i>login</i> menggunakan akun <i>User</i> .	Aplikasi menampilkan halaman beranda tanpa adanya menu Manajemen <i>User</i> dan Manajemen Kolam.	Berhasil
SK-05-1	Info <i>Profile</i>	Pengguna memilih opsi “Info <i>Profile</i> ” dari <i>dropdown</i> menu pada <i>widget profile</i> .	Halaman <i>Profile</i> pengguna berhasil diakses dan menampilkan data pengguna yang sesuai.	Berhasil
SK-06-1	Edit <i>Profile</i>	Pengguna memilih opsi	Aplikasi menampilkan	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
		“Edit <i>Profile</i> ” dari <i>dropdown</i> menu pada <i>widget profile</i> .	halaman edit <i>profile</i> dengan data <i>profile</i> yang telah terisi di setiap <i>field</i> .	
SK-06-2		Pengguna memperbarui <i>username</i> dan <i>email</i> dengan data yang valid.	Aplikasi menampilkan pesan “ <i>Profile</i> berhasil diperbarui!” dan diarahkan ke halaman <i>Profile</i> .	Berhasil
SK-06-3		Pengguna memperbarui <i>username</i> dan/atau <i>email</i> dengan data yang tidak valid atau kosong.	Aplikasi menampilkan pesan <i>error</i> .	Berhasil
SK-07-1	Status Notifikasi	Pengguna memilih opsi “Notifikasi” dari	Aplikasi menampilkan pesan "Notifikasi diaktifkan".	Berhasil

ID	Pengujian	Skenario Uji	<i>Output yang Diharapkan</i>	Status
		<i>dropdown menu pada widget profile</i> , lalu menekan <i>toggle</i> ke posisi On.		
SK-07-2		Pengguna memilih opsi “Notifikasi” dari <i>dropdown menu pada widget profile</i> , lalu menekan <i>toggle</i> ke posisi Off.	Aplikasi menampilkan dialog konfirmasi untuk mematikan <i>popup notifikasi</i> .	Berhasil
SK-07-3		Pengguna menekan tombol "Ya, Matikan" pada dialog konfirmasi.	Status notifikasi berhasil dinonaktifkan, dan <i>switch</i> berubah ke posisi Off.	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
SK-07-4		Pengguna menekan tombol "Batal" pada dialog konfirmasi.	Status notifikasi tetap aktif, dan <i>switch</i> kembali ke posisi On.	Berhasil
SK-08-1	<i>Logout</i>	Pengguna memilih opsi " <i>Logout</i> " dari <i>dropdown</i> menu pada <i>widget profile</i> .	Aplikasi menampilkan dialog konfirmasi <i>logout</i> .	Berhasil
SK-08-2		Pengguna menekan tombol "Ya" pada dialog konfirmasi.	Aplikasi melakukan proses <i>logout</i> dan kembali ke halaman <i>login</i> .	Berhasil
SK-08-3		Pengguna menekan tombol "Batal" pada dialog konfirmasi.	Proses <i>logout</i> dibatalkan, dan pengguna tetap berada di halaman sebelumnya.	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
SK-09-1	Halaman Manajemen <i>User</i>	Admin memilih menu “Manajemen <i>User</i> ” pada bilah navigasi di halaman beranda.	Aplikasi menampilkan halaman Manajemen <i>User</i> beserta daftar pengguna aplikasi dan fitur manajemen <i>user</i> seperti tambah, hapus dan edit.	Berhasil
SK-10-1	Tambah Pengguna Aplikasi	Pada halaman Manajemen <i>User</i> , Admin menekan tombol “Tambah <i>User</i> ”.	Aplikasi menampilkan halaman Tambah <i>User</i> .	Berhasil
SK-10-2		Admin memasukkan data pengguna baru seperti <i>username</i> , <i>email</i> , <i>role</i> ,	Aplikasi menampilkan pesan “ <i>User</i> berhasil ditambahkan” dan pengguna baru berhasil	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
		<i>password</i> dan <i>confirm password</i> dengan data yang valid.	ditambahkan ke dalam aplikasi dan muncul dalam daftar pengguna aplikasi.	
SK-10-3		Admin memasukkan data pengguna baru seperti <i>username</i> , <i>email</i> , <i>role</i> , <i>password</i> dan/atau <i>confirm password</i> dengan data yang tidak valid atau kosong.	Aplikasi menampilkan pesan <i>error</i> .	Berhasil
SK-11-1	Edit Pengguna Aplikasi	Pada halaman Manajemen User, Admin memilih	Aplikasi menampilkan halaman Edit Pengguna dengan data Pengguna	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
		pengguna dari daftar, lalu menekan tombol ikon tiga titik di pojok kanan atas <i>tile user</i> , kemudian pada <i>dropdown</i> yang muncul, pengguna memilih menu edit.	yang telah terisi di setiap <i>fieldnya</i> .	
SK-11-2		Admin memperbarui <i>username</i> , <i>email</i> , dan <i>role user</i> dengan data yang valid	Aplikasi menampilkan pesan “Data pengguna berhasil diperbarui!”	Berhasil
SK-11-3		Admin memperbarui <i>username</i> , <i>email</i> ,	Aplikasi menampilkan pesan <i>error</i> .	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
		dan/atau <i>role user</i> dengan data yang tidak valid atau kosong.		
SK-12-1	Hapus Pengguna Aplikasi	Pada halaman Manajemen <i>User</i> , Admin memilih pengguna dari daftar, lalu menekan tombol ikon tiga titik di pojok kanan atas <i>tile user</i> , kemudian pada <i>dropdown</i> yang muncul, pengguna memilih menu hapus.	Aplikasi menampilkan dialog konfirmasi penghapusan pengguna.	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
SK-12-2		Admin menekan tombol "Hapus" pada dialog konfirmasi.	Pengguna berhasil dihapus dan tidak muncul lagi dalam tabel daftar pengguna.	Berhasil
SK-12-3		Admin menekan tombol "Batal" pada dialog konfirmasi.	Proses penghapusan dibatalkan, dan pengguna tetap ada dalam tabel daftar pengguna.	Berhasil
SK-13-1	Tambah Kolam Baru	Pada halaman beranda, admin menekan tombol “Tambah Kolam”	Aplikasi menampilkan halaman tambah kolam.	Berhasil
SK-13-2		Admin memasukkan data <i>id</i> kolam dan nama kolam	Aplikasi menampilkan pesan “Kolam berhasil ditambahkan.”	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
SK-13-3		dengan data yang valid.		
		Admin memasukkan data <i>id</i> kolam dan/atau nama kolam dengan data yang tidak valid atau kosong.	Aplikasi menampilkan pesan <i>error</i> .	Berhasil
SK-14-1	Edit Kolam	Pada halaman beranda, Admin menekan ikon tiga titik di pojok kanan atas pada <i>tile</i> kolam, lalu memilih opsi "Edit" dari <i>dropdown</i> menu.	Aplikasi menampilkan halaman Edit Kolam dengan data Pengguna yang telah terisi di setiap <i>field</i> nya.	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
SK-14-2		Admin memperbarui data nama kolam dan status kolam dengan data yang valid.	Aplikasi menampilkan pesan “Kolam berhasil diperbarui!”	Berhasil
SK-14-3		Admin memperbarui data nama kolam dengan data yang tidak valid atau kosong.	Aplikasi menampilkan pesan <i>error</i> .	Berhasil
SK-14-4		Admin memperbarui status kolam menjadi Aktif/Non-Aktif.	Aplikasi menampilkan pesan “Kolam berhasil diperbarui!”	Berhasil
SK-15-1	Hapus Kolam	Pada halaman beranda, Admin	Aplikasi menampilkan dialog konfirmasi	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
		menekan ikon tiga titik di pojok kanan atas pada <i>tile</i> kolam, lalu memilih opsi "Hapus" dari <i>dropdown</i> menu.	penghapusan kolam.	
SK-15-2		Admin menekan tombol "Hapus" pada dialog konfirmasi.	Kolam berhasil dihapus dan tidak muncul lagi dalam daftar kolam.	Berhasil
SK-15-3		Admin menekan tombol "Batal" pada dialog konfirmasi.	Proses penghapusan dibatalkan, dan kolam tetap ada dalam daftar.	Berhasil
SK-16-1	Halaman Monitoring	Pada halaman beranda,	Aplikasi menampilkan halaman	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
		pengguna memilih kolam dari daftar kolam dengan status kolam Aktif, lalu menekan tombol “Monitoring” pada <i>tile</i> kolam yang dipilih.	Monitoring beserta <i>tile</i> grafik sensor (suhu, ph, salinitas, dan kekeruhan).	
SK-16-2		Pada halaman beranda, pengguna memilih kolam dari daftar kolam dengan status kolam Non-Aktif, lalu menekan tombol “Monitoring”	Aplikasi menampilkan pesan <i>error</i> “Monitoring untuk kolam ini dinonaktifkan karena status kolam adalah Non-Aktif.”	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
		pada <i>tile</i> kolam yang dipilih.		
SK-17-1	Menampilkan Nilai Kualitas Air	Aplikasi mengambil data sensor kualitas air (suhu, pH, salinitas, kekeruhan) secara <i>real-time</i> .	Aplikasi menampilkan parameter kualitas air secara <i>real-time</i> (suhu, pH, salinitas, kekeruhan).	Berhasil
SK-18-1	Grafik Kualitas Air	Aplikasi mengambil data kualitas air (suhu, pH, salinitas, kekeruhan) setiap 10 detik, lalu memperbarui tampilan dalam bentuk grafik.	Aplikasi menampilkan grafik yang memperbarui perubahan parameter kualitas air (suhu, pH, salinitas, kekeruhan) setiap 10 detik.	Berhasil

ID	Pengujian	Skenario Uji	<i>Output yang Diharapkan</i>	Status
SK-19-1	Halaman Pengaturan	Pada halaman monitoring, pengguna memilih <i>tile</i> grafik sensor kualitas air yang dipilih, lalu menekan tombol teks “Pengaturan Sensor <Nama_Sensor>”.	Aplikasi menampilkan halaman Pengaturan yang berisi <i>input</i> untuk mengubah nilai batas minimum dan maksimum sensor yang dipilih.	Berhasil
SK-20-1	Mengubah Batasan Parameter Sensor	Pengguna memasukkan nilai batas tertinggi dan/atau terendah parameter sensor.	Aplikasi menampilkan pesan "Batasan sensor <Nama Sensor> berhasil diperbarui."	Berhasil
SK-20-2		Pengguna memasukkan nilai batas	Aplikasi menampilkan pesan “Batas	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
		tertinggi lebih rendah dari batas terendah parameter sensor.	bawah harus lebih kecil dari batas atas.”	
SK- 21-1	Halaman Informasi Sensor	Pada halaman pengaturan sensor, pengguna menekan tombol teks "Informasi Perangkat".	Aplikasi menampilkan informasi perangkat sensor yang sesuai dengan sensor yang dipilih, termasuk nama, gambar, deskripsi, spesifikasi, dan rentang optimal penggunaan.	Berhasil
SK- 22-1	Halaman Notifikasi	Pengguna memilih menu “Notifikasi” pada bilah navigasi di	Aplikasi menampilkan halaman Notifikasi yang memuat daftar notifikasi yang relevan dengan	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
		halaman monitoring.	kolam yang dipilih di halaman Beranda.	
SK-23-1	Notifikasi Parameter Melewati Batas Aman	Aplikasi memeriksa perubahan nilai sensor pada air kolam dari sistem.	Aplikasi mengirimkan pesan notifikasi “Kualitas <Nama Sensor> air pada <Nama Kolam> berada di luar batas normal. <Nama Sensor> = <Nilai Sensor>.”	Berhasil
		Aplikasi memeriksa perubahan nilai kualitas air kolam lebih dari satu secara bersamaan dari sistem.	Aplikasi mengirimkan pesan notifikasi “Kualitas <Sensor1, Sensor2, ...> air pada <Nama Kolam> berada di luar batas normal. Sensor1 = <Nilai Sensor1>.	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
			Sensor2 = <Nilai Sensor2>.”	
SK-24-1	Notifikasi Perubahan Batasan Parameter Sensor	Aplikasi memeriksa perubahan nilai batasan maksimum dan minimum parameter sensor.	Aplikasi mengirimkan pesan notifikasi ”Batas Parameter pada <Nama Kolam> telah diubah : <Nama Sensor> : <Batasan Lama> => <Batasan Baru>.”	Berhasil
SK-25-1	Notifikasi Perubahan Data Jadwal Pemberian Pakan	Aplikasi memeriksa perubahan jadwal pemberian pakan.	Aplikasi mengirimkan pesan notifikasi “Konfigurasi pakan pada <Nama Kolam> telah diperbarui: Jadwal: <Jadwal Lama> => <Jadwal Baru>.”	Berhasil
SK-25-2		Aplikasi memeriksa	Aplikasi mengirimkan	Berhasil

ID	Pengujian	Skenario Uji	<i>Output yang Diharapkan</i>	Status
		perubahan jumlah pakan.	pesan notifikasi “Konfigurasi pakan pada <Nama Kolam> telah diperbarui: Jumlah: <Jumlah Lama> => <Jumlah Baru> gram.”	
SK-25-3		Aplikasi menerima perubahan jumlah pakan dan jadwal pemberian pakan secara bersamaan.	Aplikasi mengirimkan pesan notifikasi “Konfigurasi pakan pada <Nama Kolam> telah diperbarui: Jadwal: <Jadwal Lama> => <Jadwal Baru>. Jumlah: <Jumlah Lama> => <Jumlah Baru> gram.”	Berhasil
SK-26-1	Notifikasi Perubahan	Aplikasi memeriksa	Aplikasi mengirimkan	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
	<i>Delay Aerator</i>	perubahan <i>Delay aerator.</i>	pesan notifikasi “ <i>Delay aerator</i> untuk <Nama Kolam> telah diubah dari < <i>Delay Lama</i> > menjadi < <i>Delay Baru</i> >.”	
SK-27-1	Notifikasi Jumlah Pakan Hampir Habis	Aplikasi memeriksa isi pakan udang di dalam tabung pakan.	Aplikasi mengirimkan pesan notifikasi “Pakan pada kolam Kolam Beta hampir habis, harap segera isi ulang.”	Berhasil
SK-28-1	Melihat Notifikasi	Pada halaman notifikasi, pengguna memilih notifikasi yang ingin dilihat di	Notifikasi yang dipilih ditampilkan dengan detail lengkap dan warna judul notifikasi berubah menjadi putih	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
		daftar notifikasi.	sebagai tanda telah dibaca.	
SK-29-1	Hapus Notifikasi	Aplikasi menghapus notifikasi yang telah berumur lebih dari 7 hari.	Notifikasi yang telah berumur lebih dari 7 hari tidak muncul dalam daftar notifikasi.	Berhasil
SK-30-1	Halaman Riwayat Kualitas air	Pengguna memilih menu “Riwayat” pada bilah navigasi di halaman monitoring.	Aplikasi menampilkan halaman riwayat kualitas air yang berisi daftar laporan kualitas air.	Berhasil
SK-31-1	Pencarian Laporan Riwayat Kualitas Air	Pengguna memasukkan tanggal pada kolom <i>input</i> pencarian, dimana tanggal yang <i>diinput</i>	Aplikasi menampilkan laporan kualitas air sesuai dengan tanggal yang dimasukkan.	Berhasil

ID	Pengujian	Skenario Uji	<i>Output</i> yang Diharapkan	Status
		memiliki laporan yang sesuai di daftar laporan.		
SK-31-2		Pengguna memasukkan tanggal pada kolom <i>input</i> pencarian, dimana tanggal yang di <i>input</i> tidak memiliki laporan di daftar laporan.	Aplikasi menampilkan pesan "Laporan Riwayat Kualitas Air pada Tanggal <Tanggal Laporan> tidak tersedia."	Berhasil
SK-31-3		Pengguna menekan icon "X" di dalam kolom pencarian, setelah melakukan pencarian.	Aplikasi menghapus <i>input</i> tanggal pencarian dan menampilkan kembali seluruh daftar laporan.	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
SK-32-1	Melihat Laporan Riwayat Kualitas Air	Pada halaman riwayat, pengguna menekan tombol “Lihat Detail” pada riwayat yg dipilih.	Aplikasi menampilkan halaman Laporan Kualitas Air.	Berhasil
SK-33-1	Mengunduh Laporan Riwayat Kualitas Air	Pada halaman riwayat, pengguna menekan ikon unduh pada riwayat yg dipilih.	Aplikasi menampilkan pesan “Laporan berhasil disimpan. Lokasi file: <Lokasi File>”	Berhasil
SK-34-1	Hapus Laporan Riwayat Kualitas Air	Aplikasi menghapus laporan riwayat kualitas air yang telah berumur lebih dari 30 hari	Laporan riwayat kualitas air yang telah berumur lebih dari 30 hari tidak muncul di dalam daftar laporan.	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
		lebih dari 30 hari.		
SK-35-1	Halaman Kontrol Pakan dan <i>Aerator</i>	Pengguna memilih menu “Kontrol” pada bilah navigasi di halaman monitoring.	Aplikasi menampilkan halaman kontrol pakan dan <i>aerator</i> .	Berhasil
SK-36-1	Kontrol Otomatis Pelontar Pakan dan <i>Aerator</i>	Aplikasi memeriksa jadwal pakan. Jika waktu saat ini sesuai dengan jadwal pakan, aplikasi mengaktifkan pelontar pakan.	Pelontar pakan bekerja secara otomatis pada waktu yang telah ditentukan.	Berhasil
SK-36-2		Aplikasi memeriksa	<i>Aerator</i> berhenti otomatis saat	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
		nilai <i>Delay aerator</i> . Jika pakan telah diberikan, aplikasi menunda aktivasi <i>aerator</i> berdasarkan nilai <i>Delay</i> yang telah ditentukan.	pemberian pakan dilakukan. <i>Delay</i> sesuai dengan <i>Delay aerator</i> telah ditentukan.	
SK-37-1	Kontrol Manual Pelontar Pakan dan <i>Aerator</i>	Pengguna menekan tombol "On" pada <i>tile</i> kontrol pakan.	Aplikasi menampilkan pesan "Pakan berhasil diaktifkan" dan pelontar pakan mulai bekerja.	Berhasil
		Pengguna menekan tombol "Off" pada <i>tile</i> kontrol pakan.	Aplikasi menampilkan pesan "Pakan berhasil dinonaktifkan"	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
			pakan mulai berhenti.	
SK-37-3		Pengguna menekan tombol "On" pada <i>tile</i> kontrol <i>aerator</i> .	Aplikasi menampilkan pesan “ <i>Aerator</i> berhasil diaktifkan” dan <i>aerator</i> mulai bekerja.	Berhasil
SK-37-4		Pengguna menekan tombol "Off" pada <i>tile</i> kontrol <i>aerator</i> .	Aplikasi menampilkan pesan “ <i>Aerator</i> berhasil dinonaktifkan” dan <i>aerator</i> mulai berhenti.	Berhasil
SK-38-1	Pengaturan Jadwal Pemberian Pakan.	Pengguna memasukkan waktu yang diinginkan untuk pemberian pakan.	Aplikasi menampilkan pesan “Konfigurasi berhasil disimpan” dan menyimpan jadwal pemberian pakan yang baru.	Berhasil

ID	Pengujian	Skenario Uji	Output yang Diharapkan	Status
SK-38-2		Pengguna memilih jumlah pakan yang diinginkan	Aplikasi menampilkan pesan “Konfigurasi berhasil disimpan” dan menyimpan jumlah pakan udang yang baru.	Berhasil
SK-38-3		Pengguna memasukkan waktu dan jumlah pakan secara bersamaan.	Aplikasi menampilkan pesan “Konfigurasi berhasil disimpan” dan menyimpan waktu dan jumlah pakan udang yang baru.	Berhasil
SK-39-1	<i>Delay Aerator</i>	Pengguna memilih nilai <i>Delay</i> (waktu tunda) untuk <i>aerator</i> setelah	Aplikasi menampilkan pesan “Waktu <i>Delay aerator</i> berhasil diperbarui.”	Berhasil

ID	Pengujian	Skenario Uji	<i>Output</i> yang Diharapkan	Status
		pemberian pakan.		

Dari pengujian *Black Box Testing* yang telah dilakukan pada Tabel 4.3 diatas, seluruh skenario pengujian sebanyak 81 skenario berhasil dijalankan tanpa adanya kegagalan. Hal ini menunjukkan bahwa persentase keberhasilan pengujian mencapai 100%. Berdasarkan hasil tersebut, dapat disimpulkan bahwa aplikasi telah berjalan sesuai dengan yang diharapkan. Setiap fitur dan fungsi pada menu dalam aplikasi dapat beroperasi dengan baik dan sesuai dengan tujuan perancangan. Hasil ini juga mencerminkan bahwa aplikasi mampu memenuhi kebutuhan pengguna secara fungsional tanpa menimbulkan kesalahan selama proses pengujian.

4.1.3.2 Hasil Pengujian System *Usability Scale*

Pengujian *System Usability Scale* akan melibatkan sebanyak 30 responden untuk mengisi kuesioner. Jumlah ini dianggap cukup untuk memberikan hasil yang representatif dan dapat dipercaya sesuai dengan metode *SUS*. Responden memberikan penilaian pada 10 pernyataan dalam kuesioner *SUS*, yang mencakup aspek seperti kemudahan penggunaan aplikasi, kejelasan antarmuka aplikasi, dan kepuasan keseluruhan terhadap aplikasi monitoring dan kontrol otomatis tambak udang di Sadewa Farm. Perhitungan dilakukan dengan menggunakan persamaan 2.2 *System Usability Scale (SUS)* seperti yang telah dijelaskan pada BAB II.

Penulis mengambil sampel dari Responden 3 sebagai contoh dalam ilustrasi proses perhitungan skor *SUS*. Rincian jawaban dari responden tersebut dapat dilihat pada Tabel 4.4 di bawah ini.

Tabel 4.4 Rincian Jawaban Skor *SUS* Responden 3

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

Setelah mendapatkan skor yang diberikan oleh responden, terhadap kuesioner *System Usability Scale (SUS)*, langkah berikutnya dalam proses evaluasi adalah melakukan perhitungan skor akhir.

Sebelum sampai pada tahap perhitungan skor akhir, perlu dilakukan transformasi terhadap skor mentah yang diberikan oleh responden. Tujuannya untuk menyesuaikan interpretasi terhadap skala penilaian yang digunakan dalam kuesioner *SUS*, mengingat bahwa item-item dalam kuesioner *SUS* terdiri dari dua jenis pernyataan, yaitu pernyataan yang bersifat positif dan pernyataan yang bersifat negatif.

Pernyataan yang bersifat positif terletak pada nomor ganjil (1, 3, 5, 7, dan 9), di mana semakin tinggi skor yang diberikan oleh responden menunjukkan persepsi yang semakin baik terhadap aplikasi. Pada tahap transformasi, skor dari pernyataan positif dihitung dengan cara mengurangi setiap skor yang diberikan responden dengan angka 1. Setelah proses transformasi ini dilakukan, nilai-nilai yang dihasilkan kemudian dijumlahkan untuk mendapatkan total skor dari item pernyataan ganjil. Hasil dari skor pada item pernyataan ganjil tersebut adalah:

- Skor Ganjil = $(5 - 1) + (4 - 1) + (5 - 1) + (4 - 1) + (4 - 1)$
 $= 4 + 3 + 4 + 3 + 3$
 $= 17$

Sedangkan, pernyataan yang bersifat negatif terletak pada nomor genap (2, 4, 6, 8, dan 10), di mana semakin rendah skor yang diberikan oleh responden menunjukkan persepsi yang semakin positif. Pada tahap transformasi, dilakukan dengan cara mengurangkan skor yang diberikan responden dari angka 5. Setelah proses transformasi ini dilakukan, nilai-nilai yang dihasilkan kemudian dijumlahkan untuk mendapatkan total skor dari item pernyataan genap. Hasil dari skor pada item pernyataan genap tersebut adalah:

- Skor Genap = $(5 - 1) + (5 - 2) + (5 - 1) + (5 - 1) + (5 - 2)$
 $= 4 + 3 + 4 + 4 + 3$
 $= 18$

Selanjutnya, skor yang diperoleh dari semua item pernyataan (positif dan negatif) dijumlahkan dan kemudian dikali dengan 2.5 untuk memperoleh nilai akhir, sebagai berikut:

- Skor Akhir = $(\sum(Skor\ Ganjil + Skor\ Genap)) \times 2.5$
 $= (17 + 18) \times 2.5$
 $= (35) \times 2.5$
 $= 87.5$

Berikut ini adalah perhitungan hasil kuesioner pengujian *SUS* yang dilakukan oleh 30 responden. Data skor yang diberikan oleh masing-masing responden dapat dilihat pada Tabel 4.5 sebagai berikut.

Tabel 4.5 Skor Asli Pengujian *SUS*

Responden	Skor Asli									
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Responden 1	5	1	3	2	5	2	4	1	3	1
Responden 2	4	1	4	2	5	2	4	1	3	1
Responden 3	5	1	4	2	5	1	4	1	4	2
Responden 4	5	2	5	1	3	1	5	2	4	2
Responden 5	4	2	3	1	4	2	4	2	5	1

Responden	Skor Asli									
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Responden 6	4	1	5	2	5	3	4	2	5	1
Responden 7	5	2	4	3	4	2	5	1	4	3
Responden 8	5	3	4	1	4	1	5	1	4	1
Responden 9	4	2	5	2	4	1	4	1	3	1
Responden 10	4	3	5	1	4	2	3	2	4	2
Responden 11	4	1	5	2	5	2	5	2	5	3
Responden 12	5	2	4	1	3	2	3	2	5	2
Responden 13	4	3	5	1	4	3	4	2	4	1
Responden 14	5	1	3	2	5	1	3	1	5	1
Responden 15	5	2	4	2	5	1	5	2	4	1
Responden 16	4	1	5	2	4	2	4	1	3	2
Responden 17	4	1	4	3	4	1	4	2	4	2
Responden 18	4	2	5	1	3	1	4	1	4	1
Responden 19	5	3	4	1	4	1	3	1	5	2
Responden 20	5	1	3	2	5	2	4	1	4	1
Responden 21	4	2	4	1	4	2	5	2	5	2

Responden	Skor Asli									
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Responden 22	4	1	5	1	4	1	3	2	5	3
Responden 23	4	1	4	1	5	2	3	1	4	1
Responden 24	5	2	4	1	5	3	5	1	4	1
Responden 25	5	1	4	2	5	2	4	2	4	2
Responden 26	5	1	3	2	3	1	5	1	5	1
Responden 27	4	2	5	1	5	3	4	2	5	2
Responden 28	4	1	5	3	5	1	5	2	4	2
Responden 29	5	1	3	2	5	2	4	1	3	1
Responden 30	4	1	4	2	5	2	4	1	3	1

Setelah data skor mentah terkumpul, langkah selanjutnya adalah melakukan transformasi skor dan menghitung skor akhir untuk masing-masing responden seperti yang telah dijelaskan sebelumnya. Tabel 4.6 dibawah ini menyajikan hasil perhitungan skor akhir yang telah dihitung menggunakan metode *SUS*.

Tabel 4.6 Skor Hasil Pengujian *SUS*

Skor Hasil Hitung											Jumlah	Nilai Akhir
Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10			
3	3	4	4	3	4	4	3	3	4	35	87,5	
3	2	2	3	3	3	4	3	3	3	29	72,5	
4	4	2	3	4	3	3	4	2	4	33	82,5	
3	4	3	3	4	3	3	4	2	4	35	87,5	
4	4	3	3	4	4	3	4	3	3	35	87,5	
4	3	4	4	2	4	4	3	3	3	29	72,5	
3	3	2	4	3	3	3	3	4	4	33	82,5	
3	4	4	3	4	2	3	3	4	4	33	82,5	
4	3	3	2	3	3	4	4	3	2	35	87,5	
4	2	3	4	3	4	4	4	3	4	34	85	
3	3	4	3	3	4	3	4	2	4	32	80	
3	2	4	4	3	3	2	3	3	3	34	85	
3	4	4	3	4	3	4	3	4	2	31	77,5	
4	3	3	4	2	3	2	3	4	3	35	87,5	
3	2	4	4	3	2	3	3	3	4	33	82,5	
4	4	2	3	4	4	2	4	4	4	30	75	
4	3	3	3	4	4	4	3	3	4	34	85	
3	4	4	3	3	3	3	4	2	3	31	77,5	
3	4	3	2	3	4	3	3	3	3	31	77,5	

Skor Hasil Hitung										Jumlah	Nilai Akhir
Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10		
3	3	4	4	2	4	3	4	3	4	35	87.5
4	2	3	4	3	4	2	4	4	3	35	87.5
4	4	2	3	4	3	3	4	3	4	32	80
3	3	3	4	3	3	4	3	4	3	31	77.5
3	4	4	4	3	4	2	3	4	2	34	85
3	4	3	4	4	3	2	4	3	4	33	82.5
4	3	3	4	4	2	4	4	3	4	34	85
4	4	3	3	4	3	3	3	3	3	33	82.5
4	4	2	3	2	4	4	4	4	4	33	82.5
3	3	4	4	4	2	3	3	4	3	34	85
3	4	4	2	4	4	4	3	3	3	35	87.5
Skor Rata-Rata (Skor Akhir)										82.83	

Skor yang diperoleh dari perhitungan ini, selanjutnya diinterpretasikan ke dalam kategori-kategori tertentu yang mengacu pada standar *System Usability Scale (SUS)*, sebagaimana ditunjukkan pada Gambar 2.2. Dengan rata-rata skor sebesar 82.83 dari 30 responden, aplikasi memiliki tingkat usabilitas yang sangat baik. Dalam interpretasi *Adjective Rating*, skor 82.83 termasuk dalam kategori *Excellent*, yang menunjukkan bahwa pengguna merasa sangat puas terhadap kemudahan penggunaan dan fungsionalitas aplikasi. Berdasarkan *Grade Scale*, nilai tersebut berada pada rentang B (80–90),

yang mencerminkan bahwa aplikasi memiliki kualitas usabilitas yang baik dan hanya memerlukan sedikit peningkatan untuk mencapai tingkat yang lebih tinggi. Sementara itu, dalam *Acceptability Ranges*, skor 82.83 diklasifikasikan sebagai *Acceptable*, artinya aplikasi telah memenuhi harapan pengguna dan layak untuk digunakan. Hasil ini menunjukkan bahwa aplikasi monitoring dan kontrol otomatis tambak udang di Sadewa Farm diterima dengan baik oleh pengguna.

4.2. Analisis Hasil Penelitian

Dalam penelitian ini, peneliti menerapkan metode *Rapid Application Development (RAD)* untuk merancang dan membangun aplikasi monitoring dan kontrol otomatis pada tambak udang di Sadewa Farm. Proses pengembangan sistem dilakukan secara bertahap sesuai dengan tahapan metode *RAD*, dimulai dari perencanaan kebutuhan melalui wawancara langsung dengan pemilik tambak, kemudian dilanjutkan dengan tahapan desain antarmuka pengguna. Desain awal yang telah dikembangkan kemudian diuji secara langsung kepada pengguna untuk memperoleh umpan balik, yang menjadi dasar revisi terhadap beberapa komponen desain sebelum dilanjutkan ke tahap pengembangan sistem dan implementasi sistem.

Hasil dari evaluasi desain menunjukkan adanya dua perubahan utama pada antarmuka pengguna, yaitu pada halaman Manajemen *User* dan pada tampilan *popup* menu *profile*. Perubahan pertama dilakukan karena tampilan tabel pada halaman Manajemen *User* dinilai tidak responsif dan sulit dibaca pada perangkat *mobile*. Oleh karena itu, desain diubah menjadi model *card layout* agar lebih proporsional dan mudah diakses. Perubahan kedua dilakukan pada *popup* menu *profile*,

di mana pengguna menginginkan adanya pengaturan notifikasi secara langsung untuk menghindari gangguan saat aplikasi digunakan. Sebagai tanggapan terhadap masukan tersebut, ditambahkan fitur tombol saklar (*switch*) untuk mengatur status notifikasi.

Setelah desain mendapatkan persetujuan dari pihak Sadewa Farm, pengembangan sistem dilanjutkan dengan membangun aplikasi menggunakan Flutter untuk antarmuka pengguna dan Express.js sebagai *backend*, dengan penyimpanan data menggunakan *MongoDB*. Untuk mendukung kebutuhan monitoring kualitas air secara *real-time*, sistem juga diintegrasikan dengan *Firebase Real-time Database*.

Pada tahap implementasi, dilakukan pengujian sistem untuk memastikan seluruh fitur berfungsi dengan baik sesuai kebutuhan. Pengujian pertama dilakukan dengan menggunakan *Black Box Testing* terhadap 81 skenario yang telah disusun. Hasil pengujian menunjukkan bahwa 81 skenario berhasil dijalankan tanpa mengalami kegagalan. Dengan demikian, tingkat keberhasilan pengujian mencapai 100% yang menunjukkan bahwa seluruh fitur berjalan sesuai dengan skenario yang telah disusun. Untuk mengevaluasi tingkat kepuasan pengguna terhadap aplikasi yang telah dikembangkan, dilakukan pengujian menggunakan metode *System Usability Scale (SUS)*. Pengujian ini melibatkan 30 responden yang terdiri dari pengguna aplikasi di lingkungan tambak Sadewa Farm. Kuesioner *SUS* terdiri dari 10 pernyataan yang mengevaluasi aspek kemudahan penggunaan, efisiensi navigasi, dan kepuasan secara keseluruhan terhadap antarmuka serta alur sistem. Perhitungan skor *SUS* dilakukan dengan menjumlahkan skor dari masing-masing pernyataan, kemudian dikonversikan ke dalam skala 0–100 berdasarkan standar interpretasi *SUS*. Hasil pengujian

menunjukkan bahwa aplikasi memperoleh nilai rata-rata sebesar 82,83, yang termasuk dalam kategori “*Excellent*”. Nilai ini menunjukkan bahwa aplikasi monitoring dan kontrol otomatis pada tambak udang yang dikembangkan tidak hanya berfungsi dengan baik, tetapi juga mudah digunakan dan dipahami oleh berbagai jenis pengguna.

Dengan diterapkannya aplikasi ini dalam operasional tambak Sadewa Farm, berbagai tantangan yang sebelumnya dihadapi dalam proses budidaya dapat diminimalisir. Dari sisi pengelola, sistem ini memudahkan dalam memantau kualitas air secara *real-time* serta mengontrol perangkat seperti *feeder* dan *aerator* secara otomatis. Bagi teknisi lapangan, aplikasi ini menyajikan informasi kondisi tambak dengan cara yang lebih praktis dan cepat diakses, tanpa harus melakukan pemeriksaan secara manual. Dengan fitur-fitur yang disediakan, aplikasi ini memberikan dukungan nyata dalam mempermudah pengelolaan tambak udang secara menyeluruh.

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil pengembangan aplikasi monitoring dan kontrol otomatis untuk budidaya udang di Sadewa Farm, dapat disimpulkan bahwa metode *Rapid Application Development (RAD)* berhasil diterapkan secara efektif dalam proses perancangan hingga implementasi aplikasi. Proses pengembangan dimulai dari perencanaan kebutuhan melalui wawancara langsung dengan pemilik tambak, kemudian dilanjutkan dengan perancangan desain antarmuka yang dievaluasi oleh pengguna. Evaluasi desain menghasilkan dua perubahan utama pada halaman Manajemen *User* dan *Popup Menu Profile*. Pengembangan sistem dilakukan berdasarkan desain yang telah disetujui oleh pihak tambak Sadewa Farm. Aplikasi ini dibangun menggunakan *Flutter*, *Express.js*, dan *MongoDB*, serta diintegrasikan dengan *Firebase Real-time Database* untuk mendukung pemantauan kualitas air secara *real-time* dan pengendalian perangkat seperti *feeder* dan *aerator* secara otomatis. Sebelum implementasi sistem, dilakukan pengujian menggunakan pendekatan *Black Box Testing* dan *System Usability Scale (SUS)*.

Pengujian sistem menunjukkan tingkat keberhasilan sebesar 100% melalui metode *Black Box Testing*, dan memperoleh skor rata-rata 82,83 pada pengujian *System Usability Scale (SUS)*, yang termasuk dalam kategori “*Excellent*”. Hal ini menunjukkan bahwa aplikasi tidak hanya berfungsi secara teknis, tetapi juga diterima dengan baik oleh pengguna. Dari sisi operasional, aplikasi ini mempermudah pengelolaan tambak

udang melalui pemantauan kualitas air dan pengendalian perangkat seperti *feeder* dan *aerator* secara otomatis. serta memberikan akses informasi dengan cara yang lebih praktis dan cepat diakses, tanpa harus melakukan pemeriksaan secara manual. Dengan demikian, aplikasi ini berperan sebagai solusi praktis dalam membantu kegiatan budidaya udang di Sadewa Farm.

5.2. Saran

Terdapat beberapa saran yang dapat diberikan untuk pengembangan penelitian selanjutnya yaitu:

1. Penelitian selanjutnya disarankan untuk meninjau kembali setiap elemen *UI* dalam aplikasi, seperti menu, tombol, dan navigasi, untuk memastikan semuanya tersusun dengan baik dan mudah diakses. Penggunaan ikon yang konsisten serta label yang jelas akan membantu pengguna memahami fungsi setiap elemen dengan lebih baik, sehingga dapat meningkatkan pengalaman pengguna secara keseluruhan.
2. Penelitian selanjutnya disarankan untuk mengeksplorasi penggunaan *framework* dan *stack teknologi alternatif* seperti *React Native*, *Laravel*, atau *PostgreSQL* guna menyesuaikan dengan kebutuhan pengembangan dan performa sistem yang lebih optimal. Pendekatan ini diharapkan dapat menghasilkan solusi yang lebih adaptif terhadap dinamika dan kompleksitas dalam pengelolaan tambak yang terus berkembang.
3. Penelitian selanjutnya dapat mengembangkan aplikasi yang tidak terbatas pada aktivitas monitoring kualitas air dan kontrol *feeder* serta *aerator* saja. Pengembangan dapat mencakup penambahan

fitur kustomisasi parameter budidaya sesuai kebutuhan spesifik setiap jenis tambak, kontrol pompa sirkulasi air kolam, serta sistem pemberi pakan yang sesuai dengan siklus pertumbuhan. Dengan demikian, aplikasi dapat digunakan secara lebih fleksibel dan adaptif untuk berbagai jenis budidaya perairan, seperti budidaya ikan, udang, atau komoditas air tawar lainnya.

DAFTAR PUSTAKA

- [1] A. Zamzami, O. Fransisco, I. Irwan, dan M. I. Nugraha, “Sistem Monitoring Kualitas Air Tambak Udang Berbasis Internet of Things (IoT),” *Semin. Nas. Inov. Teknol. Terap.*, hal. 1–7, 2021.
- [2] Muttaqin, Ihsan, dan Fitria, “Peningkatan Produksi Udang Kualitas Tinggi Melalui Pengembangan Sistem Informasi Manajemen Tambak Terintegrasi Increasing Production Of High Quality Shrimp Through The Development Of An Integrated Pond Management Information System,” *Community Engagem. Emerg. J.*, vol. 4, hal. 200–206, 2023, [Daring]. Tersedia pada: <https://journal.yrpiaku.com/index.php/ceej>
- [3] Rizky Aprilia, Dadan Nur Ramadhan, dan Indrarini Dyah Irawati, “Sistem Monitoring Kualitas Air Pada Tambak Udang Vaname Di Kecamatan Kalitengah Berbasis Internet Of Things,” *e-Proceeding Appl. Sci.*, vol. 9, no. 1, hal. 306–315, 2023.
- [4] S. Anggraeni, S. Widodo, E. Wasito, dan K. Khamami, “Pendampingan Monitoring Kondisi Air Tambak Udang Varane Berbasis WEB,” *J. DIANMAS*, vol. 1, no. April, hal. 15–22, 2022, [Daring]. Tersedia pada: <http://www.jurnaldianmas.org/index.php/Dianmas/article/view/223>
- [5] D. A. Asfani *et al.*, “Alat Pemberi Pakan Udang Otomatis Portabel Berbasis Panel Surya guna Membantu Proses Budidaya Tambak Udang di Desa Tambak Plosokabupaten Lamongan,” *Sewagati*, vol. 7, no. 3, hal. 1–8, 2023, doi:

- 10.12962/j26139960.v7i3.497.
- [6] F. D. Abdillah, M. Agustini, dan Sumaryam, “Pengaruh Frekuensi Pemberian Pakan yang berbeda Terhadap Pertumbuhan Berat Mutlak udang Vaname(*Litopenaeus vannamei*) Dalam Bak Pemeliharaan,” *Juvenil*, vol. 5, no. 2, hal. 172–177, 2024.
 - [7] Alwansyah dan A. Fahrurrozi, “Implementasi Internet of Thing (Iot) Sistem Monitoring Kualitas Air Shrimp Farming Vaname Pada Aplikasi Berbasis Android,” *J. Ilm. Teknol. dan Rekayasa*, vol. 29, no. 1, hal. 71–85, 2024, doi: 10.35760/tr.2024.v29i1.11227.
 - [8] F. Wahyuni Sabran, E. Zalfiana Rusfian, M. Ilmu Administrasi dan Kebijakan Bisnis, dan F. Ilmu Administrasi, “Penggunaan Internet of Things pada eFishery untuk keberlanjutan Akuakultur di Indonesia,” *Innov. J. Soc. Sci. Res.*, vol. 3, no. 2, hal. 8142–8156, 2023, [Daring]. Tersedia pada: <https://j-innovative.org/index.php/Innovative/article/view/1359>
 - [9] A. Indra Gunawan, R. Afiful Maula, M. U. Harun Al Rasyid, dan A. Rifa'i, “Penerapan Platform Fishtech Alat Monitoring dan Kontrol Otomatis Berbasis IoT untuk Budidaya Udang di Lamongan,” *J. Appl. Community Engagem.*, vol. 2, no. 1, hal. 8–20, 2022, doi: 10.52158/jace.v2i1.266.
 - [10] D. Hariyanto *et al.*, “Implementasi Metode Rapid Application Development Pada Sistem Informasi Perpustakaan,” *J. Al-ilmi*, vol. 13, no. 1, hal. 110–117, 2021.
 - [11] Anggraini Puspita Sari, M. M. Al Haromainy, dan Ryan Purnomo, “Implementasi Metode Rapid Application

- Development Pada Aplikasi Sistem Informasi Monitoring Santri Berbasis Website,” *Decod. J. Pendidik. Teknol. Inf.*, vol. 4, no. 1, hal. 316–325, 2024, doi: 10.51454/decode.v4i1.348.
- [12] A. Almu Farrid dan R. K. Niswatin, “Testing Blackbox Untuk Kelayakan Sistem Pemilihan Siswa Unggulan,” *Stain. (Seminar Nas. Teknol. Sains)*, vol. 2, no. 1, hal. 59–66, 2023, [Daring]. Tersedia pada: <https://proceeding.unpkediri.ac.id/index.php/stains/article/view/2855>
- [13] Nasrulloh Isnain, H. Sulaiman, dan R. Rahmatika, “Pengujian Usability Pada Aplikasi Auto Reply For Messenger Menggunakan SUS,” *Explorer (Hayward)*., vol. 1, no. 2, hal. 71–80, 2021, doi: 10.47065/explorer.v1i2.97.
- [14] H. P. Ramadhan, C. Kartiko, dan A. Prasetiadi, “Monitoring Kualitas Air Tambak Udang Menggunakan Metode Data Logging,” *J. Tek. Inform. dan Sist. Inf.*, vol. 6, no. 1, hal. 102–114, 2020, doi: 10.28932/jutisi.v6i1.2365.
- [15] D. Noviandi dan P. Harahap, “Rancang Bangun Teknologi Embedded System Pemberi Pakan Ikan Berbasis Internet of Things,” *RELE (Rekayasa Elektr. dan Energi) J. Tek. Elektro*, vol. 5, no. 1, hal. 2–5, 2022, doi: 10.30596/rele.v5i1.10794.
- [16] M. Sholeh Rachmatullah, Muhammad Yazir Zain, Anang Faktchur Rachman, “Monitoring Kualitas Air Tambak udang Vaname berbasis Internet of Things,” *J. Soc. Community*, vol. 8, no. 2, hal. 116–128, 2023, [Daring]. Tersedia pada: <https://ejournal.iainata.ac.id/index.php/kabilah/article/view/291/290>

- [17] R. Dhamayanti, G. Chudra, dan A. Yohannis, “Pengembangan Aplikasi Monitoring Tambak Ikan Berbasis Internet of Things,” *Jambura J. Informatics*, vol. 5, no. 2, hal. 131–140, 2023, doi: 10.37905/jji.v5i2.19415.
- [18] J. R. Tape, A. M. Rumagi, dan S. D. S. Karouw, “Rancang Bangun Aplikasi Marketplace Pakan Ternak,” *Ranc. Bangun Apl. Marketpl. Pakan Ternak*, hal. 1–10, 2021.
- [19] M. H. R. Alauddin dan A. Putra, “Kajian Daya Dukung Lingkungan Dalam Budidaya Udang Vaname,” *J. Kelaut. dan Perikan. Terap.*, vol. 1, hal. 103, 2023, doi: 10.15578/jkpt.v1i0.12214.
- [20] R. S. Utami, Roslidar, A. Mufti, dan M. Rizki, “Sistem Kendali dan Pemantau Kualitas Air Tambak Udang Berbasis Salinitas, Suhu, dan pH Air,” *J. Komputer, Inf. Teknol. dan Elektro*, vol. 8, no. 1, hal. 43–48, 2023.
- [21] M. Z. A. Lusiana BR Ritonga, Moga Ade Sudrajat, “Manajemen Pakan Pada Pembesaran Udang Vanamei (*Litopenaeus vannamei*) di Tambak Intensif CV. Bilangan Sejahtera Bersama Feed,” vol. 47, no. 8, hal. 170–179, 2021.
- [22] A. Sihombing *et al.*, “Pengaruh Pakan Komersil dengan Kadar Protein yang Berbeda Terhadap Pertumbuhan dan Sintasan Benur Udang Windu (*Penaeus Monodon*),” *Acta Aquat. Aquat. Sci. J.*, vol. 10, no. 3, hal. 216, 2023, doi: 10.29103/aa.v10i3.12315.
- [23] S. Villamil, C. Hernández, dan G. Tarazona, “An Overview of Internet of Things,” *Telkomnika (Telecommunication Comput. Electron. Control.)*, vol. 18, no. 5, hal. 2320–2327, 2020, doi:

- 10.12928/TELKOMNIKA.v18i5.15911.
- [24] I. P. Sari, A. Novita, A.-K. Al-Khowarizmi, F. Ramadhan, dan A. Satria, “Pemanfaatan Internet of Things (IoT) pada Bidang Pertanian Menggunakan Arduino UnoR3,” *Blend Sains J. Tek.*, vol. 2, no. 4, hal. 337–343, 2024, doi: 10.56211/blendsains.v2i4.505.
 - [25] S. Somantri, G. P. Insany, dan R. R. Putra, “Perancangan Sistem Bimbingan Syarat Kecakapan Umum Pramuka Berbasis Android,” *IDEALIS Indones. J. Inf. Syst.*, vol. 6, no. 2, hal. 201–210, 2023, doi: 10.36080/idealis.v6i2.3038.
 - [26] R. S. W. Sri Dharwiyanti, “Pengantar Unified Modeling Language (UML),” *Jakarta: Bulan Bintang*, hal. 135, 2024, [Daring]. Tersedia pada: <https://books.google.co.id/books?id=0RjRNAAACAAJ>
 - [27] K. Hafidz, M. D. Irawan, dan H. D. Nawar, “Sistem Penginputan Data Bahan Pokok pada Pasar Tradisional Sumatera Utara Berbasis Website di Disperindag Sumut,” *sudo J. Tek. Inform.*, vol. 1, no. 3, hal. 98–107, 2022, doi: 10.56211/sudo.v1i3.27.
 - [28] H. Malius, Apriyanto, dan A. Ali Hakam Dani, “Sistem Informasi Sekolah Berbasis Web Pada Sekolah Dasar Negeri (Sdn) 109 Seriti,” *Indones. J. Educ. Humanit.*, vol. 1, no. 3, hal. 156–168, 2021.
 - [29] M. R. Ardonis, *Sistem Absensi Dan Penggajian Berbasis Website Menggunakan Mesin Fingerprint Pada Pt. Persada Agro Sawita Tugas*. 2020. [Daring]. Tersedia pada: <http://repository.uin-suska.ac.id/25874/1/SKRIPSI FULL TANPA BAB V.pdf>

- [30] H. Santoso, “Rekayasa Perangkat Lunak,” *Rekayasa Perangkat Lunak*, hal. 51, 2020.
- [31] I. Larasati, A. N. Yusril, dan P. Al Zukri, “Systematic Literature Review Analisis Metode Agile Dalam Pengembangan Aplikasi Mobile,” *Sistemasi*, vol. 10, no. 2, hal. 369, 2021, doi: 10.32520/stmsi.v10i2.1237.
- [32] M. I. Maliki, “Rancang Bangun Aplikasi Penjualan Grosir Sembako Pada Toko LA-RIS,” *J. Inform. dan Rekayasa Perangkat Lunak*, vol. 2, no. 3, hal. 304–311, 2021, doi: 10.33365/jatika.v2i3.1222.
- [33] A. Fau, “Pelatihan Pengenalan Dasar Framework Flutter dalam Pembangunan Aplikasi Mobile,” *J. Pengabdi. Kpd. Masy.*, vol. 01, no. 01, hal. 23–28, 2024.
- [34] Dart, “Dart Overview”, [Daring]. Tersedia pada: <https://dart.dev/overview>
- [35] D. Hadi Bachtiar, P. Paniran, dan I. M. B. Suksmadana, “Perancangan Back-end Api pada Aplikasi Mobile Fruityfit Menggunakan Framework Express JS,” *Mars J. Tek. Mesin, Ind. Elektro Dan Ilmu Komput.*, vol. 2, no. 3, hal. 107–117, 2024, [Daring]. Tersedia pada: <https://doi.org/10.61132/mars.v2i3.138>
- [36] Arlinta Christy Barus, Johannes Harungguan, dan Efren Manulu, “Pengujian Api Website Untuk Perbaikan Performansi Aplikasi Ditjenun,” *J. Appl. Technol. Informatics Indones.*, vol. 1, no. 2, hal. 14–21, 2022, doi: 10.54074/jati.v1i2.33.
- [37] E. Apriliyanto, “Comparison of Response Time Database RDBMS with NoSQL on Electronic Medical Records (EMR),”

- J. Ilm. SINUS*, vol. 21, no. 2, hal. 65, 2023, doi: 10.30646/sinus.v21i2.746.
- [38] G. Faqih Sucipto dan A. Soeharso, “Pengembangan Aplikasi E-learning Sukabaca Menggunakan Framework Express.js dan MongoDB,” *J. Pendidik. Tambusai*, vol. 7, no. 2, hal. 18757–18766, 2023.
 - [39] A. Andilala, G. Gunawan, dan K. Kirman, “Aplikasi Informasi Lowongan Pekerjaan Menggunakan Firebase Application Programming Interface Berbasis Android,” *Jtis*, vol. 4, no. 2, hal. 12–18, 2021.
 - [40] Firebase, “Firebase Documentation,” *Firebase*, 2024. <https://firebase.google.com/docs> (diakses 11 November 2024).
 - [41] A. Rifqi Yarzuq Arfani, Patmi Kasih, dan Danar Putra Pamungkas, “Pengujian Aplikasi Presensi dengan Black box Testing dengan Metode Equivalence Partitioning dan Boundary Value Analysis,” *Semin. Nas. Inov. Teknol. UN PGRI Kediri*, hal. 338, 2021.
 - [42] F. F. Zainuddin, *et al.*, “System Usability Scale (SUS) : Analisis Pengalaman Pengguna pada Portal Penerimaan Mahasiswa Baru Universitas Semarang System Usability Scale (SUS) : User Experience Analysis on The Admission Portal of Universitas Semarang,” vol. 4, no. 1, hal. 23–28, 2025.
 - [43] E. Kurniawan, N. Nofriadi, dan A. Nata, “Penerapan System Usability Scale (SUS) Dalam Pengukuran Kebergunaan Website Program Studi Di STMIK Royal,” *J. Sci. Soc. Res.*, vol. 5, no. 1, hal. 43, 2022, doi: 10.54314/jssr.v5i1.817.
 - [44] M. A. Maricar dan D. Pramana, “Usability Testing pada Sistem

- Peramalan Rentang Waktu Kerja Alumni ITB STIKOM Bali,” *J. Eksplora Inform.*, vol. 9, no. 2, hal. 124–129, 2020, doi: 10.30864/eksplora.v9i2.326.
- [45] R. Yulius, *et al.*, “Analisis Usability pada Aplikasi Amboo Mothercare Menggunakan System Usability Scale,” *JCI J. Cakrawala Ilm.*, vol. 1, no. 10, hal. 2349–2358, 2022, [Daring]. Tersedia pada: <http://bajangjournal.com/index.php/JCI>

LAMPIRAN

