

**PERANCANGAN APLIKASI PENDUKUNG TEKNOLOGI
PERIKANAN MODERN BERBASIS ANDROID**

Skripsi

**Disusun untuk memenuhi salah satu syarat
memperoleh gelar Sarjana Komputer**



**Oleh:
Gian Chiesa Maghriza
1313618021**

**PROGRAM STUDI ILMU KOMPUTER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS NEGERI JAKARTA**

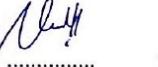
2023

LEMBAR PERSETUJUAN HASIL SIDANG SKRIPSI

LEMBAR PERSETUJUAN HASIL SIDANG SKRIPSI

PERANCANGAN APLIKASI PENDUKUNG TEKNOLOGI PERIKANAN *MODERN* BERBASIS ANDROID

Nama : Gian Chiesa Maghriza
No. Registrasi : 1313618021

	Nama	Tanda Tangan	Tanggal
Penanggung Jawab			
Dekan	: <u>Prof. Dr. Muktiningsih N. M.Si.</u>
NIP. 196405111989032001			
Wakil Penanggung Jawab			
Wakil Dekan I	: <u>Dr. Esmar Budi, S.Si., MT.</u>
	NIP. 197207281999031002		
Ketua	: <u>Dr. Ria Arafiyah, M.Si.</u>		23 - 08 - 2023
	NIP. 197511212005012004		
Sekretaris	: <u>Ari Hendarno, S.Pd., M.Kom.</u>		23 - 08 - 2023
	NIP. 198811022022032001		
Pengaji	: <u>Ir. Fariani Hermin Indiyah, M.T.</u>		23 - 08 - 2023
	NIP. 196002111987031002		
Pembimbing I	: <u>Muhammad Eka Suryana, M.Kom.</u>		23 - 08 - 2023
	NIP. 19851223200212002		
Pembimbing II	: <u>Med Izal, M.Kom.</u>		24 - 08 - 2023
	NIP. 197706152003121001		

Dinyatakan lulus ujian skripsi tanggal: 7 Agustus 2023

LEMBAR PERNYATAAN

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul **"Perancangan Aplikasi Pendukung Teknologi Perikanan Modern Berbasis Android"** yang disusun sebagai syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Ilmu Komputer Universitas Negeri Jakarta adalah karya ilmiah saya dengan arahan dari dosen pembimbing.

Simber informasi yang diperoleh dari penulis lain yang telah dipublikasikan dan disebutkan dalam teks skripsi ini, telah dicantumkan dalam Daftar Pustaka sesuai dengan norma, kaidah dan etika penulisan ilmiah.

Jika dikemudian hari ditemukan sebagian besar skripsi ini bukan hasil karya saya sendiri dalam bagian tertentu, saya bersedia menerima sanksi pencabutan gelar akademik yang saya sanding dan sanksi-sanksi lainnya sesuai dengan peraturan dan perundang-undangan yang berlaku.

Jakarta, 30 Juli 2023

Gian Chiesa Maghriza

1313618021

HALAMAN PERSEMBAHAN

Untuk Keluargaku dan Diriku Sendiri.

ABSTRAK

GIAN CHIESA MAGHRIZA. Perancangan Aplikasi Pendukung Teknologi Perikanan Modern Berbasis Android. Skripsi. Program Studi Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Jakarta. 2023. Di bawah bimbingan Muhammad Eka Suryana, M.Kom dan Med Irzal, M.Kom.

Budidaya ikan di perairan tawar merupakan salah satu aspek penting dalam sektor perikanan di Indonesia. Pencatatan data masa budidaya seperti dosis pakan, kematian ikan, grading ikan, dan sortir ikan serta hal yang berkaitan dengan kontrol kolam seperti kondisi air, data kolam merupakan hal penting dalam budidaya ikan. Saat ini pencatatan data-data tersebut masih dilakukan dengan menggunakan kertas sehingga beresiko akan kesalahan dalam perhitungan. Penelitian ini bertujuan untuk membuat aplikasi pendukung teknologi perikanan modern yang membantu pencatatan dalam melakukan aktivitas budidaya perikanan. Jenis Penelitian ini adalah Pengembangan/*Research and Development*. Informasi dan kebutuhan diperoleh dari diskusi yang dilakukan bersama pembudidaya ikan air tawar di *JFT (J Farm Technology)* serta penelusuran literatur melalui membaca berbagai jurnal yang relevan dengan topik penelitian. Diskusi menghasilkan suatu *user requirement* yang menjadi acuan fitur pencatatan yang akan diterapkan pada aplikasi pendukung teknologi perikanan modern. Proses pengembangan sistem ini menggunakan metode *Scrum* dan seluruh aplikasi yang dibuat menggunakan bahasa pemrograman *Dart* dengan *framework flutter*. Hasil akhir dari penelitian ini berupa aplikasi pendukung teknologi perikanan modern yang membantu pembudidaya dalam melakukan pencatatan aktivitas budidaya. Berdasarkan hasil uji coba *User Acceptance Test (UAT)* terdapat 6 fitur yang sudah sesuai dan 4 fitur yang belum sesuai dengan kebutuhan dilapangan, namun 4 fitur tersebut telah diperbaiki sesuai dengan kebutuhan dari pembudidaya ikan air tawar di *JFT (J Farm Technology)*.

Kata kunci: *android, aplikasi, budidaya ikan, teknologi perikanan, scrum, pencatatan*

ABSTRACT

GIAN CHIESA MAGHRIZA. Design of a Android Based Application to Support Modern Fisheries Technology. Mini Thesis. Computer Science Department, Faculty of Mathematics and Natural Sciences, State University of Jakarta. 2023. Under the guidance Muhammad Eka Suryana, M.Cs dan Med Irvan, M.Cs.

Fish farming in freshwater environments is one of the crucial aspects in the fisheries sector in Indonesia. Recording cultivation data such as feed dosage, fish mortality, fish grading, sorting, and related pond control factors like water conditions and pond data is essential in fish farming. Currently, these data recordings are still done using paper, which poses a risk of errors in calculations. This research aims to develop a supportive application for modern fisheries technology that aids in recording activities within fish farming. This research falls under the category of Research and Development. Information and requirements were obtained through discussions with freshwater fish farmers at JFT (J Farm Technology) and through literature review by studying various journals relevant to the research topic. The discussions resulted in user requirements that serve as a reference for the recording features to be implemented in the modern fisheries technology supportive application. The system development process employs the Scrum methodology, and the entire application is built using the Dart programming language with the Flutter framework. The ultimate outcome of this study is a modern fisheries technology supportive application that assists farmers in recording cultivation activities. Based on the results of the User Acceptance Test (UAT), 6 features were found to be in alignment, while 4 features did not meet field requirements initially. However, these 4 features were subsequently improved to meet the needs of freshwater fish farmers at JFT (J Farm Technology).

Keywords: android, application, cultivation of fish, fisheries, scrum, recording

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT, karena dengan rahmat dan karunia-Nya, penulis dapat menyelesaikan skripsi yang berjudul "**Perancangan Aplikasi Pendukung Teknologi Perikanan Modern Berbasis Android**".

Keberhasilan dalam menyusun skripsi ini tidak lepas dari bantuan berbagai pihak yang mana dengan tulus dan ikhlas memberikan masukan guna sempurnanya skripsi ini. Oleh karena itu dalam kesempatan ini, dengan kerendahan hati penulis mengucapkan banyak terima kasih kepada:

1. Yth. Ibu Dr. Ria Arafiyah, M.Si selaku Koordinator Program Studi Ilmu Komputer.
2. Yth. Bapak Muhammad Eka Suryana, M.Kom selaku Dosen Pembimbing I yang telah membimbing, mengarahkan, serta memberikan saran dan koreksi terhadap skripsi ini.
3. Yth. Bapak Med Irzal, M.Kom selaku Dosen Pembimbing II yang telah membimbing, mengarahkan, serta memberikan saran dan koreksi terhadap skripsi ini.
4. Direktur utama PT. Satu Teknologi Digital, bapak Roy Prasetya dan HR PT. Satu Teknologi Digital, bapak Iwan Fauzi yang senantiasa memberikan saya izin untuk dalam memenuhi keperluan skripsi yang sedang saya kerjakan.
5. Pembudidaya dari UD Jfarm yang senantiasa berpartisipasi dalam pembuatan aplikasi dalam skripsi saya.
6. Orang tua penulis yang selama ini telah mendukung dan membantu menyelesaikan skripsi ini.
7. Teman-teman Program Studi Ilmu Komputer 2018 yang telah mendukung dan membantu skripsi ini.
8. Teman-teman dari Kelompok Mahasiswa Peminat Fotografi UNJ yang telah mendukung dan menyemangati saya dalam menyelesaikan skripsi ini.
9. Teman-teman dari mabar Valorant ilmu komputer 2018 yang turut mendukung penulis dalam perngerjaan skripsi penulis.

Penulis menyadari bahwa penyusunan skripsi ini masih jauh dari sempurna karena keterbatasan ilmu dan pengalaman yang dimiliki. Oleh karenanya, kritik dan

saran yang bersifat membangun akan penulis terima dengan senang hati. Akhir kata, penulis berharap tugas akhir ini bermanfaat bagi semua pihak khususnya penulis sendiri. Semoga Allah SWT senantiasa membalaik kebaikan semua pihak yang telah membantu penulis dalam menyelesaikan skripsi ini.

Jakarta, 30 Juli 2023

Gian Chiesa Maghriza

DAFTAR ISI

LEMBAR PERSETUJUAN HASIL SIDANG	iii
LEMBAR PERNYATAAN	iv
HALAMAN PERSEMBAHAN	v
ABSTRAK	vi
ABSTRACT	vii
KATA PENGANTAR	ix
DAFTAR ISI	xi
DAFTAR GAMBAR	xiii
DAFTAR TABEL	xiv
I PENDAHULUAN	1
A. Latar Belakang Masalah	1
B. Rumusan Masalah	5
C. Pembatasan Masalah	5
D. Tujuan Penelitian	6
E. Manfaat Penelitian	6
II KAJIAN PUSTAKA	7
A. Konsep Budidaya Ikan Air Tawar	7
B. <i>Food Conversion Ratio (FCR)</i>	7
C. <i>Recirculating aquaculture system (RAS)</i>	8
D. <i>Biofloc Technology</i>	9
E. Front-end dan Back-End	11
F. Dart	12
G. Flutter	14
H. Flutter Widget	15
I. Metode Scrum	16

J.	<i>Unit Testing</i>	17
K.	<i>User Acceptance Test (UAT)</i>	18
III METODOLOGI PENELITIAN		19
A.	Deskripsi Penelitian	19
B.	Desain Penelitian	19
C.	Analisis Kebutuhan	20
D.	Perancangan Sistem	20
E.	Pengujian Sistem	23
IV HASIL DAN PEMBAHASAN		28
A.	Pembahasan	28
1.	<i>Sprint 1</i>	28
2.	<i>Sprint 2</i>	34
3.	<i>Sprint 3</i>	37
4.	<i>Sprint 4</i>	43
5.	<i>Sprint 5</i>	47
6.	<i>Sprint 6</i>	50
7.	<i>Sprint 8</i>	53
8.	<i>Sprint 8</i>	54
9.	<i>Sprint 9</i>	57
10.	<i>Sprint 10</i>	61
11.	<i>Sprint 11</i>	64
B.	Pengujian Sistem	67
1.	Unit Testing	67
2.	User Acceptance Testing	68
3.	Kesimpulan Pengujian	69
V KESIMPULAN DAN SARAN		70
A.	Kesimpulan	70
B.	Saran	70
DAFTAR PUSTAKA		72
LAMPIRAN		72
A.	Lampiran 1 Transkrip Percakapan	73
B.	Lampiran 2 code untuk sprint 1 report	74

C.	Lampiran 3 Sprint 2 report	82
D.	Lampiran 4 Code Sprint 3 report	91
E.	Lampiran 5 Code Sprint 4 report	105
F.	Lampiran 6 Code Sprint 5 report	112
G.	Lampiran 7 Code Sprint 6 report	117
H.	Lampiran 8 Code Sprint 7 report	122
I.	Lampiran 9 Code Sprint 8 report	126
J.	Lampiran 10 Code Sprint 9 report	138
K.	Lampiran 11 Code Sprint 10 report	145
L.	Lampiran 12 Code Sprint 11 report	158

RIWAYAT HIDUP**167**

DAFTAR GAMBAR

Gambar 1.1	Produksi <i>Tilapia</i> 1996-2017 (Fattah, 2020)	1
Gambar 1.2	Produksi <i>Tilapia</i> Di Pulau Jawa (KKP, 2021)	2
Gambar 1.3	Indeks Konsumsi Ikan Di Pulau Jawa (KKP, 2022)	2
Gambar 1.4	Perbandingan Tebar Padat Kolam (Setiawan, 2021)	3
Gambar 1.5	Form Pencatatan Kualitas Air	4
Gambar 2.1	Contoh Main Function	12
Gambar 2.2	Contoh Deklarasi Variable	13
Gambar 2.3	Contoh Deklarasi Variabel Secara Eksplisi	13
Gambar 2.4	Contoh Class Pada Dart	14
Gambar 3.1	Tahapan Penelitian	19
Gambar 3.2	Use Case	20
Gambar 3.3	Tahapan dan aktivitas yang dilakukan dalam metode scrum .	21
Gambar 4.1	<i>Github Projects Sprint-1</i>	29
Gambar 4.2	<i>Mock-up UI Halaman Dashboard</i>	30
Gambar 4.3	<i>Prototype Halaman Dashboard</i>	30
Gambar 4.4	<i>Struktur Direktori</i>	31
Gambar 4.5	<i>Class Diagram Sprint-1</i>	32
Gambar 4.6	<i>Screenshot dari Halaman Home yang telah selesai</i>	33
Gambar 4.7	<i>Mock-up UI List kolam, Registrasi kolam, Detail kolam, Aktivasi dan Deaktivasi kolam</i>	35
Gambar 4.8	<i>Class Diagram Fitur Koam Sprint-2</i>	35
Gambar 4.9	<i>Class Diagram Fitur Aktivasi dan Deaktivasi Sprint-2</i>	36
Gambar 4.10	<i>Output dari code pada sprint 2</i>	36
Gambar 4.11	<i>Mock-up UI pemberian pakan</i>	38
Gambar 4.12	<i>Output dari code pada rekapitulasi pakan</i>	39
Gambar 4.13	<i>Output dari code untuk halaman entry pakan</i>	39
Gambar 4.14	<i>Mock-up UI Fitur Grading</i>	44
Gambar 4.15	<i>Class Diagram Fitur Sprint-4</i>	45
Gambar 4.16	<i>Output dari code pada sprint 4</i>	45
Gambar 4.17	<i>Mock-up UI Fitur Rekapitulasi Kematian Ikan</i>	48
Gambar 4.18	<i>Class Diagram Fitur Sprint-5</i>	48

Gambar 4.19 <i>Output dari code pada sprint 5</i>	49
Gambar 4.20 <i>Mock-up UI Fitur treatment</i>	51
Gambar 4.21 <i>Class Diagram Fitur Sprint-6</i>	51
Gambar 4.22 <i>Output dari code pada sprint 6</i>	52
Gambar 4.23 <i>Mock-up UI Fitur Pencatatan Kualitas Air Harian</i>	55
Gambar 4.24 <i>Class Diagram Fitur Sprint-8</i>	56
Gambar 4.25 <i>Output dari code pada sprint 8</i>	56
Gambar 4.26 <i>Mock-up UI Fitur Pencatatan Kualitas air mingguan</i>	58
Gambar 4.27 <i>Class Diagram Fitur Sprint-9</i>	59
Gambar 4.28 <i>Output dari code pada sprint 9</i>	59
Gambar 4.29 <i>Mock-up UI Fitur Sortir</i>	62
Gambar 4.30 <i>Class Diagram Fitur Sprint-10</i>	62
Gambar 4.31 <i>Output dari code pada sprint 10</i>	63
Gambar 4.32 <i>Mock-up UI Fitur multiuser</i>	65
Gambar 4.33 <i>Class Diagram Fitur Sprint-11</i>	65
Gambar 4.34 <i>Output dari code pada sprint 11</i>	66

DAFTAR TABEL

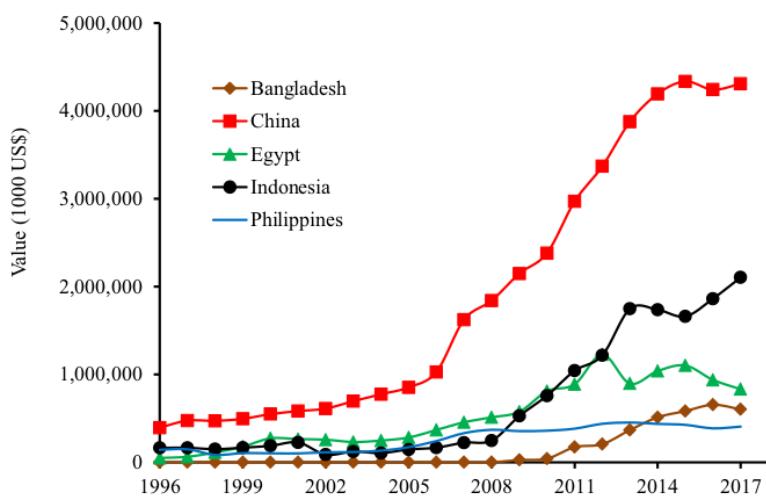
Tabel 3.1	<i>Product Backlog</i>	22
Tabel 3.2	Skenario <i>Unit Testing</i>	23
Tabel 3.3	Format <i>User Acceptance Test</i>	26
Tabel 4.1	<i>Sprint 1</i>	28
Tabel 4.2	Unit testing Halaman Dashboard.	33
Tabel 4.3	<i>Sprint 2</i>	34
Tabel 4.4	<i>Sprint 3</i>	37
Tabel 4.5	Unit testing Halaman Daftar Kolam.	40
Tabel 4.6	Unit testing Halaman Registrasi Kolam.	41
Tabel 4.7	Unit testing Halaman Detail Kolam.	41
Tabel 4.8	Unit testing Halaman Registrasi Kolam.	41
Tabel 4.9	Unit testing Halaman Aktivasi Kolam.	42
Tabel 4.10	Unit testing Halaman Deaktivasi Kolam.	42
Tabel 4.11	Unit testing Halaman Detail Masa Budidaya.	42
Tabel 4.12	Unit testing Halaman Rekapitulasi Pakan.	43
Tabel 4.13	<i>Sprint 4</i>	43
Tabel 4.14	Unit testing Halaman Rekapitulasi Grading.	46
Tabel 4.15	<i>Sprint 5</i>	47
Tabel 4.16	Unit testing Halaman Rekapitulasi Kematian.	50
Tabel 4.17	<i>Sprint 6</i>	50
Tabel 4.18	<i>Sprint 7</i>	53
Tabel 4.19	Unit testing Halaman Rekapitulasi Treatment.	54
Tabel 4.20	<i>Sprint 8</i>	54
Tabel 4.21	<i>Sprint 9</i>	57
Tabel 4.22	Unit testing Halaman Kualitas Air.	60
Tabel 4.23	<i>Sprint 10</i>	61
Tabel 4.24	Unit testing Halaman Rekapitulasi Sortir.	64
Tabel 4.25	<i>Sprint 11</i>	64
Tabel 4.26	Unit testing Halaman Awal.	67
Tabel 4.27	Unit testing Halaman Login.	67
Tabel 4.28	<i>User Acceptance Test</i>	68

BAB I

PENDAHULUAN

A. Latar Belakang Masalah

Indonesia memiliki dua sumber perikanan, yaitu tangkap laut dan budaya ikan air tawar. Berbeda dengan tangkap laut yang hanya membutuhkan skil dan modal perkapalan, budaya ikan air tawar relatif memerlukan permodalan yang lebih besar untuk dilakukan seperti lahan, infrastruktur tambak/kolam, dan juga pakan. Belum lagi budaya ikan yang tidak mudah dan memerlukan keterampilan khusus. Namun demikian bukan berarti ikan hasil tangkap menguasai sepenuhnya pasar perikanan mengingat beberapa pertimbangan yakni citara ikan air tawar dan air laut berbeda, ikan air tawar yang dapat diperjualbelikan dalam keadaan hidup, biaya dan waktu transportasi tidak memadai untuk menjangkau daerah pegunungan guna mengirim ikan hasil tangkapan. Dengan modal yang besar, tentunya hal tersebut juga diikuti oleh nilai ekonomi yang berpotensi.



Gambar 1.1: Produksi *Tilapia* 1996-2017 (Fattah, 2020)

Diantara jenis ikan air tawar yang memiliki nilai ekonomi, Lele dan Nila adalah jenis ikan yang memiliki masa panen yang cenderung singkat. Masing-masing memiliki masa panen 3 bulan serta 5 bulan. Ikan Nila (*Tilapia*) memegang 8 persen dari total produksi ikan dunia dengan produksi meningkat dari tahun ke tahun (Fattah dan El-Sayed, 2020), hal tersebut dapat dilihat pada gambar

1.1. Sementara itu data KKP menunjukkan, volume produksi *Tilapia* di pulau jawa juga memiliki tren yang sama. Kecuali DKI Jakarta yang stagnant dan Jawa Barat yang menurun, seperti yang ditunjukan oleh gambar 1.2. DKI Jakarta tidak memiliki lahan pertanian yang tidak luas sehingga dapat dimaklumi, untuk Jawa Barat penyebab penurunnya masih perlu dihuhungkan dengan ekspor-impor *Tilapia*, dan jumlah pertumbuhan rumah tangga perikanan (KKP, 2022). Ditinjau dari gambar 1.3, indeks konsumsi ikan di Jawa Barat trendnya mengalami peningkatan sehingga logika yang wajar dari turunnya jumlah produksi ikan di Jawa Barat adalah tidak terupdatenya data.

Jenis Usaha	Provinsi	Jenis Ikan	Tahun	Volume Produksi	Nilai Produksi
BUDIDAYA	BANTEN	NILA	2017	689,13	15.089.125,00
BUDIDAYA	BANTEN	NILA	2018	3.172,73	70.305.062,00
BUDIDAYA	BANTEN	NILA	2019	5.744,64	182.461.575,00
BUDIDAYA	BANTEN	NILA	2020	6.266,94	204.108.469,00
BUDIDAYA	DKI JAKARTA	NILA	2017	146,10	3.199.071,00
BUDIDAYA	DKI JAKARTA	NILA	2018	165,91	3.836.387,00
BUDIDAYA	DKI JAKARTA	NILA	2019	213,07	5.265.790,00
BUDIDAYA	DKI JAKARTA	NILA	2020	161,66	4.230.979,00
BUDIDAYA	JAWA BARAT	NILA	2017	343.361,13	7.518.235.215,00
BUDIDAYA	JAWA BARAT	NILA	2018	242.324,75	5.405.511.496,00
BUDIDAYA	JAWA BARAT	NILA	2019	294.088,87	4.675.658.397,00
BUDIDAYA	JAWA BARAT	NILA	2020	256.536,96	4.910.145.697,00
BUDIDAYA	JAWA TENGAH	NILA	2017	120.729,06	2.643.483.410,00
BUDIDAYA	JAWA TENGAH	NILA	2018	102.645,84	2.246.469.642,00
BUDIDAYA	JAWA TENGAH	NILA	2019	99.002,33	2.363.212.613,00
BUDIDAYA	JAWA TENGAH	NILA	2020	93.569,21	2.242.169.322,00
BUDIDAYA	JAWA TIMUR	NILA	2017	43.945,90	962.239.426,00
BUDIDAYA	JAWA TIMUR	NILA	2018	52.254,74	1.299.634.764,00
BUDIDAYA	JAWA TIMUR	NILA	2019	52.673,09	958.509.569,00
BUDIDAYA	JAWA TIMUR	NILA	2020	53.206,56	944.727.838,00

Gambar 1.2: Produksi *Tilapia* Di Pulau Jawa (KKP, 2021)

Provinsi	Tahun			
	2018	2019	2020	2021
BANTEN	37,41	42,94	41,29	41,74
DKI JAKARTA	45,98	50,08	48,19	48,92
JAWA BARAT	29,64	38,23	37,10	37,73
JAWA TENGAH	30,64	35,99	36,21	36,74
JAWA TIMUR	36,82	41,44	42,00	42,45

Gambar 1.3: Indeks Konsumsi Ikan Di Pulau Jawa (KKP, 2022)

Budidaya ikan air tawar di Indonesia secara umum terbagi menjadi 3 kelas yaitu *extensive*, *Semi-Intensive* dan *Intensive*. Budidaya ekstensif adalah budidaya

yang dilakukan dengan alami tanpa ada penambahan pakan buatan dari kata tradisional atau ekstensif seharusnya kita mengerti apa itu tradisional. Sementara untuk intensif dan semi-intensif, perbedaan dari keduanya terdapat pada kemampuan tebar padat ikan, dimana *Intensive aquaculture system* mampu mengakomodasi kepadatan ikan dengan luas penampang lahan yang sama. *Semi Intensive* salah satu contohnya pada sistem Tambak. Sementara *Intensive system* diwakili oleh *Biofloc Technology (BFT)* dan *Recirculating Aquaculture System (RAS)*. Ditinjau dari tingkat keahlian yang dibutuhkan *BFT* memerlukan tenaga ahli khusus.

Kualitas air yang dihasilkan pada *BFT* di atas kualitas air biasa terutama dalam 2 hal yakni, kekayaan dari bahan organik dalam air, dan kemaruan air dalam mereduksi *toxic* secara alami. *BFT* mendapatkan hal tersebut dengan bantuan *Microbacteria* dari keluarga *Bacillus sp* (Fattah dan El-Sayed, 2020). *RAS* memiliki kelebihan yaitu tidak dibutuhkan teknisi khusus kecuali saat instalasi kolam. Namun, *RAS* memiliki kemampuan tebar padat yang lebih rendah dibanding *BFT* sehingga skala produktivitasnya masih lebih rendah dibanding sistem *BFT*. Sistem *BFT* memiliki kemampuan tebar padat yang tinggi seperti yang ditunjukkan pada gambar 1.4. Namun demikian *BFT* memiliki kelemahan utama yaitu butuh tenaga ahli yang mumpuni atau rumah tangga yang mengaplikasikan *BFT* akan mengalami kegagalan. Dari data peningkatan produksi ikan hasil budidaya ikan air tawar yang telah penulis sebutkan diatas, maka penulis tertarik untuk melakukan penelitian terkait budidaya ikan air tawar untuk membantu pembudidaya dalam melakukan kegiatannya.

MINA PADI

- Extensif, padat tebar 1 ekor/m³, FCR 1 . Pakan Rp 9.500

KOLAM AIR TENANG

- Intensif. Padat tebar 10 ekor/m³ , FCR 1,5 . Pakan Rp.14.250

KERAMBA JARING APUNG

- Intensif . padat tebar 50 ekor/m³ , FCR 1,5 , Pakan Rp.14.250 / kg

KOLAM AIR DERAS

- Intensif . Padat tebar 75 - 100 ekor/m³ . FCR 1.7 . Pakan Rp. 16.150 / kg

SISTEM BIOFLOC / COKLAT

- Intensif, Padat tebar 100 – 150 ekor/m³. FCR 1.1 .Pakan Rp. 11.500

Gambar 1.4: Perbandingan Tebar Padat Kolam (Setiawan, 2021)

Dalam penelitian ini, penulis telah melakukan wawancara dengan UD Jfarm sebagai salah satu lembaga budidaya yang bergerak di bidang budidaya ikan air tawar modern berbasis *BFT* yang beralamat di desa Pamagersari, kecamatan Jasinga, Kabupaten Bogor yang menjadi narasumber. Narasumber mengungkapkan permasalahan yang dihadapi selama proses budidaya di antarnya tenaga ahli bidang *BFT* di UD Jfarm melakukan pemeriksaan kualitas air yang ketat setiap hari. Pengukuran kualitas air ini dilakukan untuk setiap kolam budidaya yang ada seperti yang ditunjukkan pada Gambar 1.5. Hal ini tentunya berimplikasi pada pengelolaan arsip formulir tersebut perlu dilakukan dengan benar dan rapi agar data yang dikumpulkan dapat dievaluasi sebagai dasar pertimbangan *treatment* yang akan dilakukan pada suatu kolam budidaya. Jika hal ini tidak dilakukan, maka berpotensi pada kegagalan proses budidaya berbasis *BFT*. Potensi kegagalan tersebut juga diperbesar dengan metode pencatatan yang dilakukan masih berbasis kertas. Atas dasar masalah tersebut, penulis mengusulkan pengembangan aplikasi pendukung teknologi perikanan modern yang dapat membantu pembudidaya ikan air tawar, dalam hal ini UD Jfarm melakukan pencatatan data yang dibutuhkan selama musim budidaya berlangsung.

Pemeliharaan :			AKTIVITAS KOLAM BUDIDAYA BIOFLOC IKAN NILA		
No	PARAMETER	MINGGU KE :		Jumlah el/or:	
		HARI KE	TGL	JUMLAH	Kolam :
PERSIAPAN MEDIA BIOFLOC					
1	ISI AIR	Cm			
2	RAPORIT	gram			
3	AERASI	ceklis			
4	GARAM IKAN	kg			
5	DOLOMIT	gram			
6	MOLASE ENCER	ml			
7	ULTIMA-PRO	ml			
TEBAR IKAN PADA PAGI HARI					
1	AKLIMATISASI	ceklis			
2	SAMPLING	ceklis			
3	SIZE GRAM/EKOR	gram			
4	JUMLAH TOTAL kg	kg			
PERAWATAN HARIAN PAGI HARI JAM 07:00 - SELESAI					
1	GANTI BATU UDARA	ceklis			
2	CEK IKAN MATI	ekor			
3	BUANG ENDAPAN	5 detik			
4	UKUR DO/OKSIGEN	angka			
5	UKUR PH PAGI	angka			
6	DOLOMIT	gram			
7	ULTIMA-PRO	ml			
8	MAKAN PAGI 8:00	gram			
9	MAKAN SIANG 13:00	gram			
10	CEK PH SORE	angka			
11	MAKAN SORE 17:00	gram			
12	CEK IKAN MUNTAH	ceklis			
PERAWATAN MINGGUAN PAGI HARI JAM 07:00 - SELESAI					
1	PUASA IKAN	ceklis			
2	CEK PH	angka			
3	CEK NITRIT/NO2	angka			
4	CEK FLOC	%			
5	PENGENCERAN AIR	%			
6	MOLASE/TEPUNG	ml/gram			
7	AKTIVASI PROBIOTIK	liter			
8	CEK OKSIGEN	mg/l			
9	BUANG ENDAPAN	ceklis			
10	SAMPLING/KILOGRAM	ekor			
11	PORSI MAKAN SEHARI	gram			

Catatan : Aktivasi dibuat sehari sebelum hari pemberian dan langsung habis.
Aktivasi ULTIMA-PRO diberikan ke kolam sebanyak 500ml/kubik air.
Pakan ikan sebaiknya dfermentasi , ideal 24 jam kedap udara.

Gambar 1.5: Form Pencatatan Kualitas Air

Sudah banyak peneliti di Indonesia yang berkontribusi dalam berbagai teknologi *Aqua Culture*. (Supriati dan Rizki, 2018) meneliti dan mengembangkan Sistem Informasi Akuntansi Budidaya Perikanan berbasis *SAK EMKM* dan *Android*. Aplikasi yang dikembangkan diperuntukkan untuk Kelompok Tani yang tersusun atas Ketua, Sekretaris, Bendahara, dan Seksi Sarana Produksi perikanan. Aplikasi ini memfasilitasi sejumlah fitur seperti transaksi jual-beli, kas keluar-masuk, dan pembukuan. Penelitian yang sejenis dilakukan oleh (Widhiastika dkk., 2021), aplikasi hasil penelitian memfasilitasi user untuk melakukan transaksi jual beli perikanan. Aplikasi yang dibuat dinamakan dengan *FO-KLIK*. Dikembangkan dengan *Android* yang dipasangkan dengan *Firebase*. Selain transaksi jual beli, *FO-Klik* turut menyediakan fasilitas diskusi dan informasi nilai gizi dari suatu produk item perikanan. Namun tidak dilengkapi dengan manajemen keuangan seperti yang dilakukan oleh (Supriati dan Rizki, 2018). Selain dari penelitian diatas, perancangan aplikasi terkait budidaya ikan air tawar juga dapat dilakukan berdasarkan metode atau sistem budidaya yang digunakan. Adapun skripsi Andri Rahmanto yang berjudul "Perancangan Arsitektur Aplikasi Budidaya Perikanan Modern pada *Backend* yang Bertanggung Jawab Melayani Transaksi *Query Webservice* Dengan Menggunakan Teknologi *Flask Microservice*" dimana penulis akan menggunakan *Backend* yang telah buat dalam bentuk *REST API*.

Berdasarkan latar belakang yang telah dijelaskan, Peneliti mengusulkan pengembangan aplikasi manajemen budidaya perikanan. Aplikasi ini berpotensi meningkatkan jumlah ikan yang dipanen dan menurunkan angka kematian sehingga mampu menurunkan harga jual ikan dengan adanya stok melimpah. Dengan demikian, dapat terwujud ketahanan pangan wilayah. Aplikasi ini diharapkan dapat membantu petani meningkatkan efisiensi dari setiap masa budidaya yang dijalani.

B. Rumusan Masalah

Dari uraian latar belakang di atas, perumusan masalah pada penelitian ini ialah "Bagaimana perancangan aplikasi pendukung teknologi perikanan modern berbasis android?"

C. Pembatasan Masalah

Pembatasan masalah pada penelitian ini yaitu:

1. Aplikasi pendukung teknologi perikanan modern dikembangkan sampai pada fitur pencatatan dan manajemen masa budidaya.
2. Penelitian ini hanya merangjang bagian *frontend* dari aplikasi pendukung teknologi perikanan modern

D. Tujuan Penelitian

Membuat aplikasi pendukung teknologi perikanan modern yang memudahkan pencatatan data budidaya perikanan.

E. Manfaat Penelitian

1. Bagi penulis

Memperluas pengetahuan tentang teknologi perikanan modern, menambah pengalaman dalam *programming*, memperoleh gelar sarjana di bidang Ilmu Komputer, serta menjadi media untuk penulis dalam mengaplikasikan ilmu yang didapatkan dari kampus.

2. Bagi Program Studi Ilmu Komputer

Penelitian ini dapat menjadi pembuka untuk penelitian di masa depan, dan dapat memberikan panduan bagi mahasiswa program studi Ilmu Komputer tentang perancangan aplikasi teknologi perikanan modern.

3. Bagi Universitas Negeri Jakarta

Menjadi evaluasi akademik program studi Ilmu Komputer dalam penyusunan skripsi sehingga dapat meningkatkan kualitas pendidikan dan lulusan program studi Ilmu Komputer di Universitas Negeri Jakarta.

BAB II

KAJIAN PUSTAKA

A. Konsep Budidaya Ikan Air Tawar

Budidaya adalah kegiatan memproduksi dan mengembangkan biota (organisme) dalam lingkungan yang terkendali untuk mendapatkan keuntungan. Budidaya ikan air tawar merupakan kegiatan yang dilakukan untuk meningkatkan produktivitas perairan, khususnya ikan air tawar. Kegiatan budidaya dimaksudkan untuk memperbanyak, menumbuhkan dan meningkatkan kualitas biota perairan itu sendiri untuk menghasilkan keuntungan. Sementara budidaya perikanan modern merupakan metode budidaya yang menggabungkan teknologi terbaru dan ilmu pengetahuan untuk menciptakan lingkungan yang optimal bagi pertumbuhan dan kesehatan ikan (Bangkit, 2016).

Akuakultur (budidaya ikan) merupakan salah satu subsektor yang diharapkan dapat mewujudkan misi mensejahteraan perikanan. Akuakultur tingkat rendah berkontribusi pada kesejahteraan petani ikan untuk menjamin ketersediaan pangan, gizi dan kesehatan di daerah. Budidaya perikanan modern menggunakan pelet dan konsetrat sebagai sumber pakan, selain itu juga diterapkan berbagai treatment kedalam sistem budidaya untuk membantu proses budidaya ikan. Contoh metode budidaya perikanan modern yaitu RAS dan Biofloc.

B. Food Conversion Ratio (FCR)

FCR (Feed Conversion Ratio) pada budidaya ikan adalah suatu ukuran yang menyatakan rasio jumlah pakan yang dibutuhkan untuk menghasilkan 1 kg daging ikan. FCR juga sering digunakan untuk mengetahui kualitas pakan yang diberikan terhadap pertumbuhan ikan. Informasi mengenai FCR ikan yang dibudidayakan sangat penting karena berkaitan dengan efisiensi pakan dan efisiensi musim budidaya. Menurut (Effendie, 1997) FCR ikan didapatkan melalui rumus berikut.

$$FCR = \frac{Berat\ Total\ Ikan\ Panen(Kg)}{Jumlah\ Pakan(Kg)}$$

Informasi mengenai FCR ikan yang dibudidayakan sangat penting karena berkaitan dengan efisiensi pakan. Efisiensi pakan berfungsi mengukur tingkat

penggunaan input, yakni pakan dan output berupa bobot daging ikan. Semakin kecil nilai efisiensi pakan, berarti pakan yang diberikan sudah baik. Efisiensi pakan berhubungan dengan pertambahan berat, konsumsi pakan, dan konversi pakan. Pertambahan berat dapat dihitung dengan mengurangi bobot badan ikan saat panen dengan bobot ikan pada awal penebaran.

Sementara itu, konversi pakan merupakan pembagian antara berat badan yang dicapai pada bulan berlangsung dengan konsumsi pakan pada bulan tersebut. Konversi pakan didapatkan dengan membagi total pakan yang dikonsumsi dengan total hasil produksi.

Efisiensi berkaitan erat dengan biaya yang harus Anda keluarkan selama budidaya. Apabila efisiensi pakan tidak bagus, kemungkinan besar biaya pakan yang dikeluarkan juga besar. Padahal, pakan merupakan komponen penting dan besar dalam usaha budidaya. Semakin besar biaya pakan, akan semakin besar juga biaya produksi yang dibutuhkan.

C. *Recirculating aquaculture system (RAS)*

RAS merupakan sistem budidaya yang menggunakan air daur ulang yang pertama kali diperkenalkan di Amerika Serikat pada awal tahun 1960 dan mulai diterapkan sejak tahun 1990-an. Teknologi RAS pada saat itu menjadi solusi atas permasalahan pencemaran organik sungai dari tempat budidaya bersamaan dengan permintaan benih ikan salmon yang tinggi yang dibutuhkan sepanjang waktu (kontinu). Kualitas suatu perairan merupakan syarat penting yang dapat mempengaruhi kelangsungan hidup perkembangan, pertumbuhan, dan tingkat produksi ikan. Lingkungan yang baik sangat diperlukan untuk kelangsungan hidup organisme akuatik. Sistem ini telah banyak dikembangkan dan iterapkan di beberapa negara maju, seperti Amerika, Israel, Singapura, German serta Norwegia selama kurun waktu 20-30 tahun ini (Fadhil dkk., 2010).

Teknologi RAS menawarkan sebuah alternatif teknologi budidaya melalui perbaikan kualitas air dan penggunaan kembali (re-use). Penggunaan RAS secara intensif terbukti dapat mengurangi secara signifikan konsumsi air dan konsentrasi nutrien melalui perbaikan dan pengembangan teknologi secara berkelanjutan (Thesiana dan Pamungkasi, 2015). RAS dapat digunakan untuk mengontrol beberapa parameter kualitas air penting seperti oksigen terlarut, karbondioksida, amonia, nitrit, nitrat, pH, salinitas, dan padatan tersuspensi. Hal ini memungkinkan

terciptanya kondisi pemeliharaan yang baik untuk pertumbuhan dan pemanfaatan pakan yang lebih optimal (Dalsgaard dkk., 2015).

D. *Biofloc Technology*

Teknologi bioflok merupakan teknologi penggunaan bakteri baik heterotrof maupun autotrof yang dapat mengonversi limbah organik secara intensif menjadi kumpulan mikroorganisme yang berbentuk flok, kemudian dapat dimanfaatkan oleh ikan sebagai sumber makanan. Di dalam flok terdapat beberapa organisme pembentuk seperti bakteri, plankton, jamur, alga, dan partikel tersuspensi yang memengaruhi struktur dan kandungan nutrisi bioflok, namun komunitas bakteri merupakan mikroorganisme paling dominan dalam pembentukan flok dalam bioflok.

Agregat bioflok memiliki rentang ukuran partikel yang luas, mulai dari yang mikroskopis hingga yang lebih besar dari 1 mm. Bahkan organisme yang lebih besar seperti copepoda dan nematoda dapat memakan flok dan menjadi bagian yang tidak terpisahkan dari beberapa agregat. Kepadatan flok dengan berat basah biasanya hanya sedikit lebih besar dari 1 g/mL, jadi agregat akan tenggelam perlahan-lahan dan relatif mudah dipelihara suspensinya. Dengan porositas hingga 99 persen (ruang kosong), nutrisi, oksigen, dan produk limbah mudah dipertukarkan di antara bagian dalam flok dan air di sekitarnya, dan ini ditingkatkan oleh pencampuran umum dalam sistem bioflok(Samocha, 2019).

Mikroorganisme bioflok bervariasi antar sistem dan juga dalam sistem yang sama dari waktu ke waktu mengidentifikasi fluktuasi dalam komposisi taksonomi bakteri, mikroalga, ragi, dan mikroorganisme lain dalam flok dari sistem bioflok ikan nila. Di antara bakteri dan ragi taksa adalah Aeromonas spp., Vibrio spp., Enterobacter sp., Nitrospira sp., Bacillus spp., Sphingomonas sp., Pseudomonas spp., Microthrix sp., Nitrobacter sp., Micrococcus sp., Alcaligenes sp., dan Rhodotorula sp. Bakteri biasanya mendominasi bioflok di sistem akuakultur. Tidak hanya berlimpah (hingga 100 juta bakteri/mL), tetapi juga menunjukkan keragaman yang tinggi. terdapat gambaran menyeluruh pembahasan banyak faktor yang menentukan komposisi flok, diantaranya adalah suhu, salinitas, pH, penyinaran, intensitas pencampuran vertikal, dan jenis karbon organik yang tersedia untuk metabolisme bakteri.

Laju perkembangan flok bisa ditingkatkan dengan cara menambahkan organik karbon untuk merangsang pembentukan flok. Perkembangan dipengaruhi

oleh berbagai faktor, yang terutama adalah suhu, oksigen terlarut, pH, beban organik, cahaya, dan pencampuran. (Samocha, 2019) juga menjelaskan mengenai faktor-faktor tersebut:

1. Agregat lebih besar dan lebih padat di tempat yang suhunya lebih tinggi dan oksigen terlarut lebih tinggi.
2. Pencampuran yang intens mengganggu agregat dan mengurangi ukuran flok rata-rata.
3. Laju pemompaan tinggi melalui lubang kecil mengurangi ukuran flok.
4. Oksigen terlarut lebih rendah mendukung filamen bakteri, kemungkinan karena rasio permukaan-ke-volume yang tinggi.
5. pH mempengaruhi flok secara langsung, setiap spesies memiliki range pH optimal masing-masing dan berhubungan langsung dengan alkalinitas, karbon anorganik, dan amonia.
6. Muatan organik tinggi berpromosi lebih cepat berkembang.
7. Cahaya mempengaruhi kelimpahan organisme fotoautotrofik (yaitu, cyanobacteria, ganggang hijau, diatom, dinoflagellata, rhodophyta, dll.) dalam flok.

Di antara keunggulan bioflok dalam akuakultur adalah nilai nutrisinya yang tinggi, perannya dalam meningkatkan kualitas air, dan efek probiotiknya pada ikan:

1. Bioflok sebagai pakan

Bioflok memiliki kualitas nutrisi yang mirip dengan makanan yang dimakan ikan liar di habitat alaminya. Mempertahankan kepadatan flok yang sesuai di seluruh siklus tanaman dapat mengurangi kebutuhan akan pakan yang diformulasikan (Avnimelech, 2009) yang biasanya menyumbang setidaknya setengah dari biaya produksi dalam akuakultur tradisional. Kuantitas dan kualitas bahan organik disimpan oleh bakteri akhirnya menentukan nilai gizi flok. Bahan organik yang disimpan ini tergantung pada jumlah dan jenis organik karbon yang tersedia untuk pertumbuhan bakteri. intinya adalah bahwa "bioflok adalah apa yang dimakannya". Jika substrat organik yang tepat disediakan, kemudian floc akan menyimpan senyawa berkualitas tinggi itu, maka hal tersebut akan berkontribusi terhadap kebutuhan nutrisi ikan.

2. Bioflok dan kualitas air

Di luar nilai nutrisinya, bakteri bioflok dapat dikelola untuk meningkatkan kualitas air. Ini dapat diklasifikasikan menurut caranya memperoleh yaitu, autotrof dan heterotrof. Baik organisme autotrofik dan heterotrofik yang mengisi agregat bioflok meningkatkan kualitas air dengan asimilasi atau transformasi senyawa nitrogen anorganik terlarut (amonia, nitrit, nitrat) yang berbeda derajat, hal ini berbahaya bagi ikan. Untuk tujuan ini, sistem yang didominasi bioflok dapat dikelola mendukung bakteri autotrofik, bakteri heterotrofik, atau kombinasi keduanya dalam sistem mixotrophic. Setiap pilihan memiliki perbedaan implikasi terhadap kualitas air.

3. Bioflok dan respon immune

Beberapa jenis ikan memiliki sistem kekebalan yang labil. Ini berarti bahwa mereka tidak memiliki antibodi-antigen spesifik mekanisme untuk menanggapi patogen baru. Populasi mikroba dalam sistem bioflok, Namun, mungkin berperan dalam mengaktifkan sistem kekebalan non spesifik mereka, menghasilkan pertahanan

Pada sistem akuakultur dengan teknologi bioflok, air media kultur hanya sekali dimasukkan dalam wadah, dan digunakan sampai panen. Penambahan air hanya untuk pengganti penguapan dan pengontrolan kepadatan bioflok (Avnimelech, 2009). Dibanding sistem resirkulasi yang sangat kompleks, sistem kultur dengan teknologi bioflok hanya menggunakan satu wadah, yakni wadah kultur. Penguraian bahan organik oleh bakteri dan mikroorganisme pengurai, sampai pada pemanfaatan hasil penguraian oleh mikroalga dan mikroorganisme yang tumbuh, terjadi dalam wadah secara seimbang dengan kepadatan organisme kultur yang sangat tinggi. Pengontrolan kualitas air terjadi dalam wadah kultur itu sendiri, oleh sistem bioflok yang sudah berjalan dalam wadah kultur. Sistem ini sangat murah, sederhana ramah lingkungan dan memiliki produktifitas yang sangat tinggi (Taw, 2012).

E. Front-end dan Back-End

Frontend adalah bagian yang dilihat dan dilihat oleh pengguna untuk berinteraksi dengannya, seperti menu, formulir kontak, dll. Dalam konteks web, front end dibuat dengan menggunakan HyperText Markup Language (HTTP), Cascading Style Sheets (CSS), dan juga JavaScript. Sehingga, suatu URL bisa bekerja dan menampilkan situs website dengan baik. Sementara dalam aplikasi

mobile, front end dibuat dengan menggunakan widget untuk menampilkan suatu fitur untuk berkomunikasi dengan pengguna. Oleh karena itu, front end juga bisa disebut sebagai client-side.

Di sisi lain, backend biasanya terdiri dari tiga bagian: server, aplikasi, dan DB. Bagian belakang teknologi biasanya terdiri dari bahasa seperti PHP, Ruby, Python, dll. Pada bagian back end, semua hal yang dibuat di dalam front end ataupun sistem dan server dibalik dibuatnya situs website atau aplikasi bisa bekerja sebagaimana mestinya. Mereka juga sering disebut dengan server-side.

F. Dart

Bahasa pemrograman Dart merupakan bahasa pemrograman general-purpose yang dirancang oleh Lars Bak dan Kasper Lund. Bahasa pemrograman ini dikembangkan sebagai bahasa pemrograman aplikasi yang dapat dengan mudah untuk dipelajari dan disebarluaskan. Bahasa pemrograman Dart dapat digunakan secara bebas oleh para developer, karena bahasa ini dirilis secara open-source oleh Google di bawah lisensi BSD. Bahasa pemrograman Dart merupakan bahasa pemrograman berbasis class dan berorientasi terhadap obyek dengan menggunakan sintaks bahasa pemrograman (Kelvin dan Dian, 2021). Berikut adalah contoh program sederhana dari Dart guna lebih memahami bahasa pemrograman tersebut:

1. *Main Function*

Setiap aplikasi mempunyai Main Function, berikut adalah contoh Main Function yang akan menampilkan teks pada console. Untuk menerapkannya, bisa digunakan print() function.

```
void main() {
    print('Hello, World!');
}
```

Gambar 2.1: Contoh Main Function

2. Deklarasi *Variables*

Berikut adalah contoh deklarasi Variable pada dart:

```
var name = 'Bob';
```

Gambar 2.2: Contoh Deklarasi Variable

Pada gambar 2.2 dapat diketahui bahwa terdapat variable yang bernama name dengan bentuk String bernilai "Bob". Variable akan disimpulkan secara implisit dalam bentuk String meskipun tidak dideklarasi secara eksplisit, namun jika dibutuhkan untuk mendeklarasikan variabel secara eksplisit, maka menggunakan cara deklarasi secara eksplisit seperti pada gambar 2.3.

```
String name = 'Bob';
```

Gambar 2.3: Contoh Deklarasi Variabel Secara Eksplisi

3. *Classes*

Dart adalah bahasa berorientasi objek dengan Class dan pewarisan berbasis mixin. Setiap objek adalah turunan dari Class, dan semua Class kecuali Null turun dari Objek. Pewarisan berbasis mixin berarti meskipun setiap Class (kecuali Top-Class) memiliki tepat satu superclass, Class dapat digunakan kembali dalam beberapa hierarki Class. Metode Extendsi adalah cara untuk menambahkan fungsionalitas ke Class tanpa mengubah Class atau membuat subclass. Dapat Dilihat pada gambar 2.4 yaitu contoh sebuah Class pada bahasa pemrogramman Dart.

```

class Spacecraft {
  String name;
  DateTime? launchDate;

  // Read-only non-final property
  int? get launchYear => launchDate?.year;

  // Constructor, with syntactic sugar for assignment to members.
  Spacecraft(this.name, this.launchDate) {
    // Initialization code goes here.
  }

  // Named constructor that forwards to the default one.
  Spacecraft.unlaunched(String name) : this(name, null);

  // Method.
  void describe() {
    print('Spacecraft: $name');
    // Type promotion doesn't work on getters.
    var launchDate = this.launchDate;
    if (launchDate != null) {
      int years = DateTime.now().difference(launchDate).inDays ~/ 365;
      print('Launched: ${launchYear} ($years years ago)');
    } else {
      print('Unlaunched');
    }
  }
}

```

Gambar 2.4: Contoh Class Pada Dart

G. Flutter

Flutter telah diresmikan sejak tahun 2015 oleh developer Dart yaitu Sky. Eric Seidel (Direktur untuk Flutter diGoogle). Flutter adalah cross-platform framework yang dibuat oleh Google untuk membangun berbagai aplikasi seperti android mobile, iOS mobile, web, dan desktop. sedangkan menurut (Napoli, 2019) Flutter adalah kerangka antarmuka pengguna portabel (UI) Google untuk membangun aplikasi modern, asli, dan reaktif untuk iOS dan Android. Framework ini menggunakan widgets untuk membuat UI dan Dart sebagai bahasa pemrograman untuk mengembangkan aplikasinya. Widget dapat diibaratkan dengan permainan Lego yang bisa menambahkan berbagai macam jenis bongkahan plastik kecil dan mengubah tampilan UI sesuai dengan yang diinginkan oleh developer.

Flutter mempunyai dua macam widget untuk pengembang aplikasi pakai, yaitu Material Design dan Cupertino. Material Design adalah bahasa design yang dibuat oleh Google, design ini sama dengan design yang dipakai pada Android. Cupertino atau dengan sebutan lain gaya iOS adalah bahasa design yang dipakai oleh iOS. Flutter mempunyai lebih banyak widget Material Design daripada widget Cupertino, walaupun demikian, pada OS perangkat yang berbeda, widget bisa dipakai secara cross platform. Flutter memiliki berbagai macam keuntungan dibanding framework lain diantaranya:

1. Tingkatkan produktivitas

Dalam mengembangkan aplikasi berbasis Android dan iOS, Anda hanya membutuhkan satu basis kode. Ini membuat Anda lebih menghemat waktu dan tim.

2. Tingkatkan produktivitas

Tampilan desain yang sangat bagus. Flutter menyediakan widget cantik yang dapat disesuaikan dengan mudah untuk membuat aplikasi terlihat lebih menarik. Jadi, dengan menggunakan Flutter, developer dari setiap level keahlian akan lebih mudah membuat aplikasi terlihat lebih baik.

3. Mudah untuk dipelajari

Flutter adalah framework yang mudah dipelajari. Selain itu, Flutter juga memungkinkan developer aplikasi untuk membangun aplikasi tanpa menggunakan banyak kode seperti pada framework lainnya.

4. Performa kerangka kerja yang hebat

5. Pembuatan hemat biaya

Membuat dua aplikasi (Android dan iOS) dengan menggunakan satu codebase tentunya dapat lebih menghemat biaya pengembangan aplikasi.

6. Tersedia dalam berbagai pilihan aplikasi IDE (Integrated Development Environment)

Anda dapat memilih untuk mengembangkan aplikasi (Flutter) menggunakan Android Studio, atau dengan VS Code.

7. Dokumentasi lengkap dan komunitas aktif.

Anda dapat membuka dokumentasi melalui situs resmi Flutter. Selain itu, Anda juga bisa bergabung dengan komunitas untuk berdiskusi dengan sesama pengguna Flutter.

H. Flutter Widget

Flutter menggunakan konsep widget untuk membuat tampilannya. Widget adalah komponen UI yang membangun aplikasi Flutter. Setiap membuat UI dalam Flutter akan menciptakan berbagai macam pohon widgets . Berikut beberapa informasi mengenai widget:

1. Widget digunakan untuk membuat UI, seperti: baris, kolom, stack, card, form, dan padding.

2. Pada Widget, developer dapat melakukan styling, seperti: tipe font, ukuran, berat, warna, batas, dan bayangan.
3. Widget dapat berupa kumpulan bentuk dan formulir.

Diantara widget-widget tersebut, ada beberapa widget yang sering digunakan dan diterapkan dalam berbagai macam aplikasi seperti: CardView, PageView, Bottom Navigation View. Untuk mengetahui lebih lanjut tentang widget-widget pada flutter.

I. Metode Scrum

Metode Scrum diciptakan oleh Jeff Sutherland dan tim pengembangnya pada awal 1990-an. Sutherland, Viktorov, Blount dan Puntikov menggambarkan Scrum sebagai “Proses pengembangan perangkat lunak *Agile* yang dirancang untuk menambah energi, fokus, kejelasan, dan transparansi bagi tim proyek yang mengembangkan sistem perangkat lunak” (Sutherland dkk., 2007). Proses Scrum mematuhi prinsip-prinsip pendekatan *Agile*. Prinsip pendekatan *Agile* berfokus pada memuaskan pelanggan melalui pengiriman awal perangkat lunak yang berharga; membolehkan perubahan kebutuhan; kolaborasi antara pengembang dan pelaku bisnis; perangkat lunak yang berfungsi (sebagai ukuran kemajuan); menjaga desain tetap sederhana; dan secara berkala, membuat tim merenungkan bagaimana menjadi lebih efektif selama proses pengembangan.

Scrum menawarkan cara yang dapat disesuaikan untuk bekerja pada proyek yang berbeda yang memiliki berbagai persyaratan dan Scrum memiliki keunggulan seperti pemilihan persyaratan atau spesifikasi kebutuhan yang fleksibel untuk *sprint* dan tidak ada prosedur khusus yang harus diikuti. Prinsipnya adalah bekerja secara iteratif hingga mencapai waktu yang ditentukan dan dapat memenuhi kebutuhan konsumen. Scrum menerapkan metode ilmiah empirisme. Scrum menggantikan pendekatan algoritmik terprogram dengan pendekatan heuristik, dengan menghormati orang dan pengaturan diri untuk menghadapi ketidakpastian dan memecahkan masalah yang kompleks. Unit dasar Scrum adalah tim kecil dari beberapa orang. Tim Scrum terdiri dari satu Scrum Master yang bertanggung jawab untuk membangun Scrum sebagaimana didefinisikan dalam Panduan Scrum, satu Product Owner yang bertanggung jawab untuk memaksimalkan nilai produk yang dihasilkan dari pekerjaan Tim Scrum, dan Developers yang berkomitmen untuk menciptakan aspek apa pun dari Increment yang dapat digunakan setiap Sprint. Dalam Tim Scrum, tidak ada sub-tim atau hierarki. Ini adalah unit profesional yang

kohesif yang berfokus pada satu tujuan pada satu waktu, Sasaran Produk. Aktivitas-aktivitas yang ditentukan digunakan di Scrum agar terciptanya keteraturan dan meminimalkan kebutuhan akan rapat yang tidak ditentukan dalam Scrum. Semua aktivitas dibatasi oleh waktu. Setelah Sprint dimulai, durasinya tetap dan tidak dapat dipersingkat atau diperpanjang. Aktivitas yang tersisa dapat berakhir setiap kali tujuan aktivitas tercapai, memastikan jumlah waktu yang tepat dihabiskan tanpa membiarkan pemborosan dalam proses. Aktivitas-aktivitas yang terdapat pada Scrum adalah Sprint, Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective.

Penjelasan lebih lanjut mengenai Scrum dapat dilihat pada jurnal (Sutherland dan Schwaber, 2020) yang berjudul "Scrum Guides". Adapun penelitian Andri Rahmanto menggunakan metode scrum dalam pengembangan penelitiannya yang berjudul "Perancangan Arsitektur Aplikasi Budidaya Perikanan Modern pada Backend yang Bertanggung Jawab Melayani Transaksi Query Webservice Dengan Menggunakan Teknologi Flask Microservice".

J. Unit Testing

Unit testing merupakan salah satu tipe pengujian perangkat lunak dimana setiap fungsi atau komponen dari perangkat lunak diuji. Tujuan dari unit testing adalah untuk memastikan fungsi pada perangkat lunak sudah berjalan sesuai dengan ekspektasi (Hamilton, 2022). Menurut Rosa dan Shalahuddin (2013: 277), "Unit testing berfokus pada pengujian unit terkecil (komponen perangkat lunak atau modul) dari desain perangkat lunak. Semua fungsi pada perangkat lunak diuji untuk memastikan bahwa input dan output unit sesuai dengan yang diinginkan".

Pressman (2012) menyebutkan bahwa teknik yang dilakukan pada unit testing adalah "berfokus pada setiap unit perangkat lunak (misalnya komponen, kelas, atau objek konten dari aplikasi atau web) seperti yang diterapkan dalam kode program". Kode program dikaji apakah terdapat kesalahan. Kesalahan pada kode program dapat diidentifikasi dengan menggunakan White-Box Testing. Sedangkan menurut Rosa dan Shalahuddin (2013), "White-Box Testing (pengujian kotak putih) merupakan pengujian perangkat lunak atau aplikasi dari segi desain dan kode program untuk mengetahui apakah perangkat lunak mampu menghasilkan fungsi-fungsi, masukan, dan keluaran yang sesuai dengan spesifikasi".

Proses unit testing memastikan fungsi-fungsi pada aplikasi yang telah

dikembangkan peneliti memenuhi persyaratan, dapat berjalan dengan baik, dan memiliki input serta output sesuai yang diinginkan. Berdasarkan definisi di atas dapat disimpulkan bahwa unit testing merupakan salah satu tipe pengujian fungsi atau unit pada perangkat lunak untuk memastikan apakah perangkat lunak mampu untuk menghasilkan input dan output sesuai dengan spesifikasi yang telah ditentukan.

K. *User Acceptance Test (UAT)*

Menurut Perry, William E, (2006) User Acceptance Test (UAT) yaitu pengujian untuk verifikasi apakah fungsi pada sistem telah berjalan dengan kebutuhan, pengujian dilakukan oleh user dimana user tersebut adalah staff/karyawan perusahaan yang langsung berinteraksi dengan sistem. Menurut Black, acceptance testing pada umumnya menunjukkan bahwa sistem telah memenuhi persyaratan-persyaratan tertentu. Pada pengembangan software dan hardware komersial, acceptance test biasanya disebut juga "alpha tests" (yang dilakukan oleh pengguna in-house) dan "beta tests" (yang dilakukan oleh pengguna yang sedang menggunakan atau akan menggunakan sistem tersebut).

Hadji, Haryono, Rahayu, (2019) menyebutkan bahwa pelaksanaan UAT terdapat pada akhir proses pengujian saat sistem siap digunakan. Tujuan utama UAT adalah untuk memvalidasi apakah sistem diterima atau ditolak, memenuhi spesifikasi sistem, dan mengembangkan perangkat lunak yang mampu memenuhi kebutuhan user. Proses UAT memastikan bahwa aplikasi pendukung teknologi perikanan modern, sudah layak diujikan secara masif, memenuhi harapan pengguna, dan bekerja seperti yang diharapkan. Dari definisi yang telah dipaparkan dapat disimpulkan bahwa UAT adalah pengujian pada akhir proses yang dilakukan oleh pengguna pada sebuah sistem untuk memastikan fungsi-fungsi yang terdapat pada sistem tersebut telah berjalan dengan baik dan sesuai dengan kebutuhan pengguna

BAB III

METODOLOGI PENELITIAN

A. Deskripsi Penelitian

Aplikasi yang akan dibuat pada penelitian ini adalah aplikasi pendukung teknologi perikanan modern. Aplikasi berfungsi untuk mencatat setiap kegiatan yang dilakukan oleh petani ikan air tawar selama masa budidaya berlangsung. Selain hal tersebut, aplikasi ini juga dapat mengukur *Food Conversion Ratio* dari setiap musim budidaya yang diselesaikan oleh petani.

B. Desain Penelitian



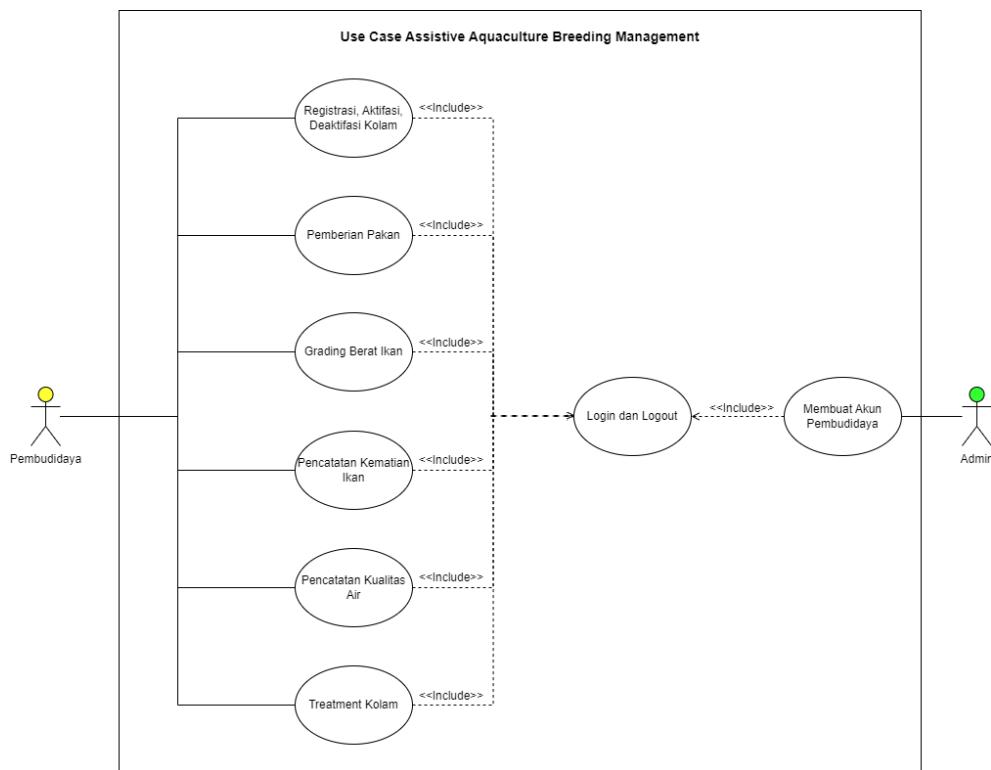
Gambar 3.1: Tahapan Penelitian

Agar penelitian berjalan lebih terstruktur dan mudah, maka penulis memerlukan desain penelitian. Desain penelitian merupakan proses dan tahapan yang dilakukan penulis dalam melakukan pembuatan aplikasi pendukung teknologi

perikanan modern dengan metode scrum. Tahapan penelitian yang akan dilaksanakan penulis dalam perancangan aplikasi dapat dilihat pada gambar 3.1.

C. Analisis Kebutuhan

Berdasarkan uraian pada lampiran A yang berisi tentang wawancara analisis kebutuhan fitur untuk aplikasi, prioritas fitur pada aplikasi pendukung teknologi perikanan modern akan berfokus pada penerapan dari backend yang telah dibuat oleh Andri Rahmanto kedalam aplikasi. Berikut adalah *usecase* yang telah didefinisikan berdasarkan hasil wawancara.

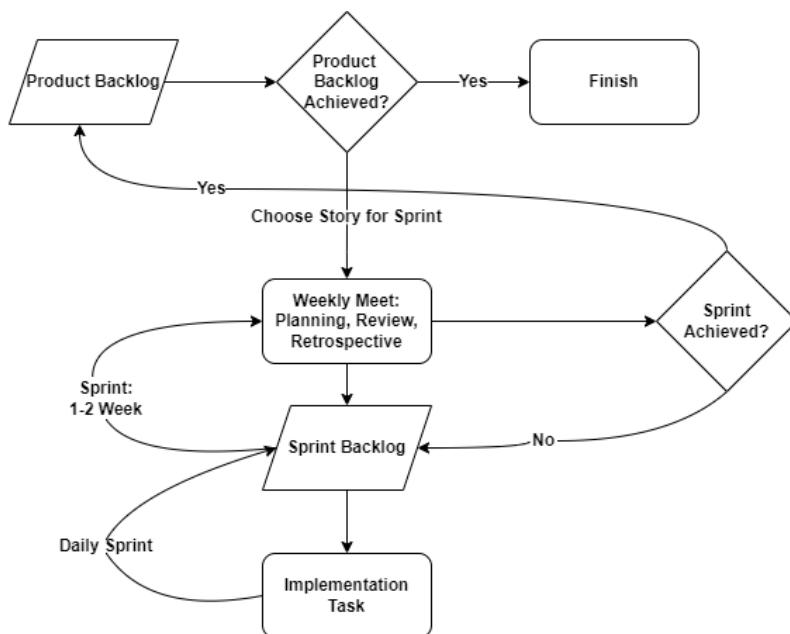


Gambar 3.2: Use Case

D. Perancangan Sistem

Metode perancangan sistem pada penelitian ini sesuai dengan komponen yang ada di dalam metode Scrum. Komponen-komponen tersebut terdiri dari *product backlog*, *sprint backlog*, *sprint*, *daily scrum*, dan *pengujian sistem*. Aktivitas-aktivitas yang ditentukan digunakan di Scrum agar terciptanya keteraturan.

Semua aktivitas dibatasi oleh waktu. Setelah Sprint dimulai, durasinya tetap dan tidak dapat dipersingkat atau diperpanjang. Aktivitas yang tersisa dapat berakhir setiap kali tujuan aktivitas tercapai, memastikan jumlah waktu yang tepat dihabiskan tanpa membiarkan pemborosan dalam proses. Aktivitas-aktivitas yang terdapat pada Scrum adalah Sprint, Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective. tahapan dan aktivitas tersebut dapat dilihat pada gambar 3.3.



Gambar 3.3: Tahapan dan aktivitas yang dilakukan dalam metode scrum

1. *Product Backlog*

Tabel 3.1 merupakan tabel *product backlog* aplikasi pendukung teknologi perikanan modern yang telah dibuat berdasarkan diskusi dengan *Scrum Master*. Setiap *item* dari *product backlog* ini akan diimplementasikan pada *sprint* dari awal hingga selesai.

Tabel 3.1: Product Backlog

No	Story	Sprint	Status
1	Membuat halaman dasboard	1	
2	Membuat fitur registrasi kolam dan aktifasi Musim budidaya	2 dan 3	
3	Membuat fitur pemberian pakan	3	
4	Membuat fitur grading berat ikan perkolam	4	
5	Membuat fitur pencatatan kematian	5	
6	Membuat fitur treatmet kolam	6 dan 7	
7	Membuat fitur pencatatan kualitas air harian	8	
8	Membuat fitur pencatatan kualitas air mingguan	9	
9	Membuat fitur perpindahan ikan antar kolam	10	
10	Membuat fitur Multi User	11	

Berdasarkan tabel 3.1, *Product Backlog* pada penelitian ini terdiri dari 3 komponen, yaitu *story*, *sprint*, *status*. *Story* merupakan task besar yang nantinya akan dipecah lagi menjadi *task-task* yang kecil di dalam *Sprint*. Komponen *sprint* pada tabel ini menandakan pada *sprint* berapa *story* tersebut akan dilaksanakan. Komponen *status* menjelaskan apakah *story* tersebut sudah selesai atau belum.

2. Sprint Backlog

Sprint backlog adalah daftar task yang perlu dikerjakan ataupun yang sudah dikerjakan pada *sprint*. Di dalam *sprint backlog*, berbagai *task* kecil dibuat. Dengan Sprint Backlog, seluruh anggota tim dapat melihat perkembangan dari setiap pekerjaan.

3. Sprint

Setelah dilakukan perencanaan pada Sprint Backlog, maka penggerjaan sprint sudah dapat dimulai dan harus mengikuti jadwal penggerjaan yang telah disepakati bersama tim. Dalam penelitian ini, interval sprint yang digunakan adalah satu sampai dua minggu.

4. Sprint Review dan Sprint Retrospective

Sprint review, dan *sprint retrospective* dilakukan pada setiap awal pekan yaitu di hari Selasa melalui *voice call* menggunakan platform discord dan juga

tatap muka secara langsung. Pada awal pekan ini akan dilakukan evaluasi mengenai perkembangan proses pembuatan aplikasi maupun hambatan yang terjadi selama penggeraan di setiap *sprint*.

E. Pengujian Sistem

Pada tahap ini peneliti akan melakukan uji aplikasi pendukung teknologi perikanan modern menggunakan dua jenis pengujian yaitu unit testing dan User Acceptance Test (UAT). Pengujian unit testing dilaksanakan oleh tim internal developer untuk memastikan fungsi-fungsi pada aplikasi yang telah dikembangkan dapat berjalan dengan baik. Sedangkan UAT dilaksanakan oleh pengguna dalam hal ini product owner yaitu UD Jfarm untuk mengetahui apakah aplikasi sudah sesuai kebutuhan dan layak digunakan.

1. Unit Testing

Skenario pada unit testing dibuat berdasarkan product backlog. Adapun skenario dari unit testing yang akan dilaksanakan terdapat pada tabel yang terdapat pada tabel dibawah ini.

Tabel 3.2: Skenario *Unit Testing*

Skenario <i>Unit Testing</i>	
Uji Fitur	Skenario Pengujian
Halaman Awal	Saat aplikasi dibuka akan muncul splash screen yang menampilkan logo yang menandakan aplikasi sedang loading
	Setelah loading selesai maka akan ditampilkan halaman login
Login	Ketika mengisi form login dengan data yang sesuai kemudian menekan submit, maka akan masuk ke halaman dashboard
	Ketika mengisi form login dengan data yang tidak sesuai kemudian menekan submit, maka akan menampilkan pesan kesalahan
Halaman Dashboard	Saat ikon profil ditekan maka akan muncul halaman profil lembaga farm

Skenario Unit Testing	
Uji Fitur	Skenario Pengujian
	Saat tombol kolam ditekan maka akan muncul halaman yang menampilkan list kolam
Daftar Kolam	Saat ikon (+) ditekan maka akan menampilkan halaman registrasi kolam
	Saat card kolam ditekan akan menampilkan halaman detail kolam
Registrasi Kolam	ketika mengisi form registrasi kolam dengan data yang sesuai dan menekan submit, maka kolam baru akan ditambahkan
Detail Kolam	ketika tombol kualitas air ditekan maka akan menampilkan halaman kualitas air
	ketikan menekan salah satu list dari masa budidaya akan menampilkan halaman detail masa budidaya
	ketika status kolam tidak aktif atau panen dan menekan tombol start budidaya, maka akan menampilkan halaman aktivasi kolam
	ketika status kolam aktif dan menekan tombol aktif, maka akan menampilkan halaman deaktivasi kolam
Aktivasi Kolam	ketika mengisi form aktivasi kolam dengan data yang sesuai dan menekan submit, maka musim budidaya baru akan ditambahkan dan status kolam menjadi aktif
Deaktivasi Kolam	ketika mengisi form deaktivasi kolam dengan data yang sesuai dan menekan submit, maka musim budidaya yang barjalan akan berhenti status kolam menjadi tidak aktif
Kualitas Air	Ketika memilih musim budidaya dan maka akan ditampilkan list pengontrolan kualitas air kolam
	Ketika menekan list data pengontrolan kualitas air, maka akan ditampilkan detail pengontrolan kualitas air kolam
	Saat ikon (+) ditekan maka akan menampilkan halaman entry pengontrolan kualitas air kolam

Skenario Unit Testing	
Uji Fitur	Skenario Pengujian
	ketika mengisi form pengontrolan kualitas air kolam dengan data yang sesuai dan menekan submit, data pengontrolan kualitas air akan ditambahkan
Detail Masa Budidaya	Ketika tombol rekapitulasi pakan ditekan, maka akan ditampilkan halaman rekapitulasi pakan
	Ketika tombol rekapitulasi kematian ikan ditekan, maka akan ditampilkan halaman rekapitulasi kematian ikan
	Ketika tombol rekapitulasi grading ikan ditekan, maka akan ditampilkan halaman rekapitulasi grading ikan
	Ketika tombol treatment ditekan, maka akan ditampilkan halaman treatment kolam
	Ketika tombol sortir ditekan, maka akan ditampilkan halaman sortir kolam
Rekapitulasi Pakan	Ketika menekan list data rekapitulasi pakan, maka akan ditampilkan detail rekapitulasi pakan
	Ketika menekan list data rekapitulasi pakan, maka akan ditampilkan detail rekapitulasi pakan
	Saat tombol entry pakan ditekan maka akan menampilkan halaman entry pakan
	ketika mengisi form rekapitulasi pakan dengan data yang sesuai dan menekan submit, data rekapitulasi pakan akan ditambahkan
Rekapitulasi Kematian	Ketika menekan list data rekapitulasi pakan, maka akan ditampilkan detail rekapitulasi pakan
	Saat tombol entry kematian ditekan maka akan menampilkan halaman entry kematian
	ketika mengisi form rekapitulasi kematian dengan data yang sesuai dan menekan submit, data rekapitulasi kematian akan ditambahkan
Rekapitulasi Grading	Ketika menekan list data rekapitulasi grading, maka akan ditampilkan detail rekapitulasi grading

Skenario Unit Testing	
Uji Fitur	Skenario Pengujian
	Ketika menekan list data rekapitulasi grading, maka akan ditampilkan detail rekapitulasi grading
	Saat tombol entry grading ditekan maka akan menampilkan halaman entry grading
	ketika mengisi form rekapitulasi grading dengan data yang sesuai dan menekan submit, data rekapitulasi grading akan ditambahkan
Halaman Treatment	Ketika menekan list data treatment, maka akan ditampilkan detail treatment
	Saat ikon (+) ditekan maka akan menampilkan halaman entry treatment
	ketika mengisi form treatment dengan data yang sesuai dan menekan submit, data treatment akan ditambahkan
Halaman Sortir	Ketika menekan list data sortir, maka akan ditampilkan detail sortir
	Saat ikon (+) ditekan maka akan menampilkan halaman entry sortir
	‘

2. User Acceptance Test (UAT)

User Acceptance Test dibuat berdasarkan fitur-fitur yang dapat diakses oleh pengguna pada product backlog. Adapun format dari UAT yang akan dilaksanakan terdapat pada tabel 3.3 dimana penulis menggunakan format yang digunakan oleh (Keat dan Chuah, 2018) sebagai referensi.

Tabel 3.3: Format User Acceptance Test

User Acceptance Test					
No	Acceptance Requirements	Kesesuaian			
		SS	S	TS	STS
1	Fitur login sudah sesuai dengan kebutuhan pembudidaya				

<i>User Acceptance Test</i>					
No	Acceptance Requirements	Kesesuaian			
		SS	S	TS	STS
2	Fitur dasboard sudah sesuai dengan kebutuhan pembudidaya				
3	Fitur registrasi, aktivasi dan deaktivasi sudah sesuai dengan kebutuhan pembudidaya				
4	Fitur pemberian dan rekapitulasi pakan sudah sesuai dengan kebutuhan pembudidaya				
5	Fitur grading berat ikan sudah sesuai dengan kebutuhan pembudidaya				
6	Fitur pencatatan kematian sudah sesuai dengan kebutuhan pembudidaya				
7	Fitur treatment kolam sudah sesuai dengan kebutuhan pembudidaya				
8	Fitur pencatatan kualitas air harian sudah sesuai dengan kebutuhan pembudidaya				
9	Fitur pencatatan kualitas air mingguan sudah sesuai dengan kebutuhan pembudidaya				
10	Fitur sortir ikan sudah sesuai dengan kebutuhan pembudidaya				

BAB IV

HASIL DAN PEMBAHASAN

A. Pembahasan

Aplikasi pendukung teknologi perikanan modern, dirancang dengan menggunakan metode Scrum. Pada metode Scrum, proses pengembangan sistem dilakukan secara bertahap yang disebut dengan Sprint. Pada penelitian ini terdapat sembilan Sprint dimana satu putaran Sprint memiliki durasi selama dua minggu. Pada tiap awal pekan, dilakukan perencanaan Sprint Backlog berdasarkan Product Backlog telah disepakati. Adapun laporan setiap Sprint yang dilakukan pada proses pengembangan sistem adalah sebagai berikut:

1. *Sprint 1*

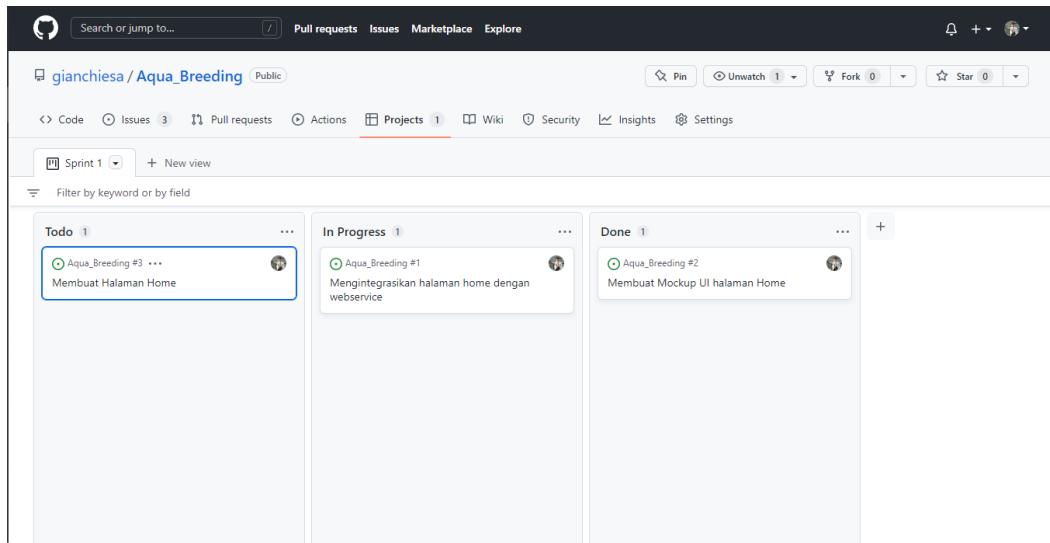
Sprint-1 dilakukan sepekan pada tanggal 23 Agustus 2022 sampai dengan 30 Agustus 2022. Story pertama pada *product backlog* yaitu membuat halaman dashboard dipecah menjadi beberapa *task* sebagai berikut.

Tabel 4.1: *Sprint 1*

No	Story	Task	Status
1	Membuat Halaman Dashboard	Membuat <i>Mock-up UI</i> halaman dashboard	selesai
2		Menerapkan struktur direktori dan membuat class diagram	selesai
3		Menerapkan <i>Mock-up UI</i> halaman dashboard ke Flutter	selesai
4		Mengintegrasikan halaman home ke <i>webservice</i>	selesai

Pada sprint pertama ini story yang di pilih untuk di uraikan pada sprint kali ini adalah membuat halaman dashboard. Tujuan dari *sprint-1* ini adalah membuat halaman home dan mengintegrasikan halaman tersebut dengan *webservice* yang sudah dibuat oleh penelitian Andri Rahmanto. Kendala yang dialami penulis pada

sprint kali ini adalah diperlukannya waktu training terhadap bahasa pemrograman dan framework yang digunakan dalam melakukan pengembangan aplikasi, penulis menggunakan *GitHub Projects* seperti pada gambar 4.1.

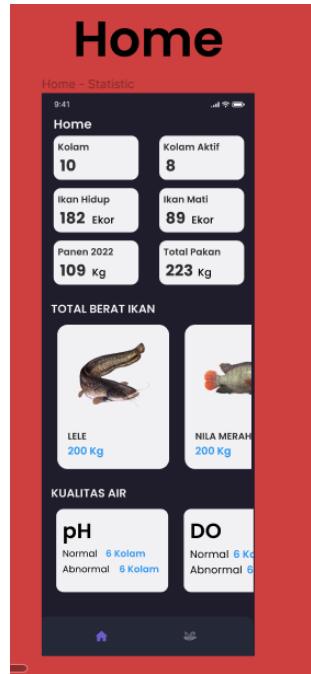


Gambar 4.1: Github Projects Sprint-1

Terdapat 3 kolom pada *project* yang dibuat, yaitu *to do*, *in progress*, dan *completed*. Setiap *task* yang perlu dikerjakan akan ditulis dan dimasukkan ke dalam kolom *to do*, selanjutnya *task* yang sedang dikerjakan akan dipindahkan ke kolom *in progress*, dan jika task sudah selesai akan dipindahkan ke kolom *completed*. Berikut merupakan hasil dari pengerjaan yang dilakukan selama *sprint 1*.

1. Membuat Mock-up UI Halaman Dashboard

Pembuatan konten dan fitur yang terdapat pada *mock-up UI* halaman home dilakukan berdasarkan persetujuan product owner dan scrum master pada meeting sebelumnya. Mock-up UI dibuat menggunakan platform figma.



Gambar 4.2: Mock-up UI Halaman Dashboard

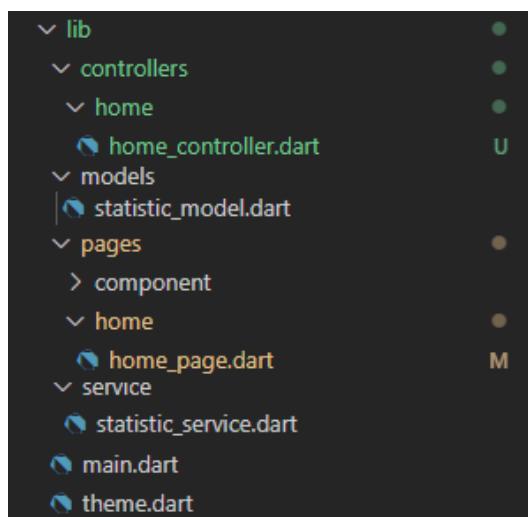
Lalu selanjutnya dibuat prototype agar product owner dan scrum master dapat lebih mengerti bagaimana konten dan fitur tersebut berjalan saat akan dikembangkan seperti pada gambar 4.3.



Gambar 4.3: Prototype Halaman Dashboard

2. Menetapkan Struktor Direktori

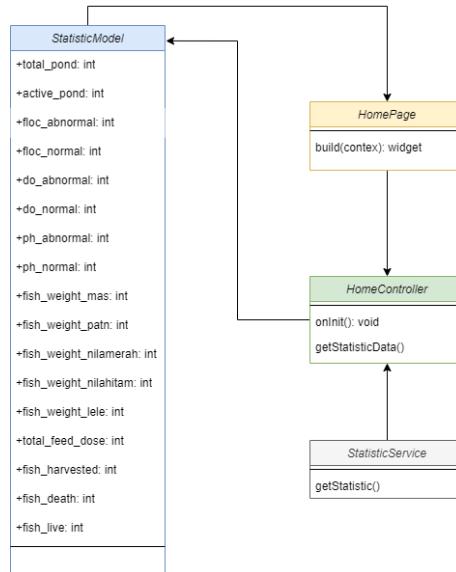
Sebelum menerapkan Mockup-UI kedalam code, penulis membuat atau menetapkan struktur direktori file agar memudahkan penulis untuk melakukan proses pengembangan untuk kedepannya. Terdapat 4 direktori yang penulis buat pada penelitian kali ini, yaitu models, controllers, pages, dan services. Direktori models akan memuat file yang berfungsi sebagai model class, lalu pada direktori controller berisi file controller-controller yang akan digunakan pada suatu halaman/fitur untuk memanipulasi/mengolah data yang didapat dari services. Direktori services berisi file yang digunakan untuk berinteraksi dengan dunia luar seperti pemanggilan API dan membuat network request, untuk direktori pages memuat file UI halaman.



Gambar 4.4: Struktur Direktori

3. Class Diagram

Class Diagram menggambarkan kelas-kelas yang akan dipakai oleh sistem. Umumnya terdapat 3 kelas pada setiap module yaitu class model, controller, dan view. Pada sprint-1 penelitian kali ini penulis membuat 4 class yaitu model, view , controller, dan service.



Gambar 4.5: Class Diagram Sprint-1

Pada gambar 4.5 class HomePage sebagai view menginisiasi controller dari interaksi dengan user, lalu controller akan memanggil mengfetch data dari service class dan memasukan data tersebut ke model class, setelah itu model class akan memberi tahu sistem jika data sudah terupdate dan akan mengirimkan data tersebut kembali ke view.

4. Menerapkan Mockup-UI Halaman Dashboard kedalam code flutter

Setelah *mock-up UI Halaman Dashboard* hingga prototype dibuat, akan dilakukan pengimplementasian *mock-up UI* ke dalam aplikasi menggunakan flutter. Pada lampiran 2 terdapat source code dari implementasi halaman home yang dikelompokan berdasarkan section.

5. Mengintegrasikan Halaman Dashboard dengan webservice

Sebelumnya, setiap data pada halaman home masih berupa data dummy sehingga perlu diintegrasikan dengan webservice agar aplikasi dapat berjalan dengan data yang asli. pada lampiran 6 terdapat langkah-langkah untuk mengintegrasikan halaman dashboard dengan webservice.

6. Sprint 1 Review dan Sprint 2 Planning

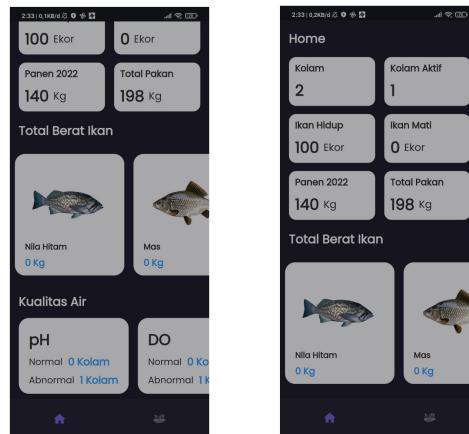
Sprint 1 diakhiri dengan melakukan weekly meeting pada hari selasa dengan agenda melakukan review dan testing terkait hasil sprint 1 dan melakukan planning untuk sprint 2 dengan rincian:

(a) *Review dan Testing hasil dari sprint 1*

Telah dilakukan review dan testing oleh penulis selaku developer dengan Scrum Master. Setelah dilakukan testing, Scrum Master menyimpulkan bahwa fitur pada halaman dashboard telah berjalan dengan baik.

Tabel 4.2: Unit testing Halaman Dashboard.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Ketika mengisi form login dengan data yang sesuai kemudian menekan submit, maka akan masuk ke halaman dashboard	✓		Diterima
Saat tombol kolam ditekan maka akan muncul halaman yang menampilkan list kolam	✓		Diterima



Gambar 4.6: Screenshot dari Halaman Home yang telah selesai

(b) *Sprint Planning untuk Sprint 2*

Planning untuk sprint 2 yakni membuat fitur registrasi kolam dan aktifasi musim budidaya pada aplikasi *Assistive Aquaculture Breeding Management*.

2. Sprint 2

Sprint-2 dilakukan sepekan pada tanggal 30 Agustus 2022 sampai dengan 6 September 2022. *Story* kedua pada *product backlog* yaitu membuat fitur registrasi kolam dan aktivasi musim budidaya dipecah menjadi beberapa *task* sebagai berikut.

Tabel 4.3: Sprint 2

No	Story	Task	Status
1	Membuat fitur registrasi kolam dan aktivasi musim budidaya	Membuat <i>Mock-up UI</i> halaman list kolam, registrasi kolam, detail kolam, aktivasi dan deaktivasi kolam	selesai
2		Menerapkan struktur direktori dan membuat class diagram	selesai
3		Menerapkan <i>Mock-up UI</i> halaman list kolam, registrasi kolam, detail kolam, aktivasi dan deaktivasi kolam	selesai
4		Mengintegrasikan halaman list kolam, registrasi kolam, detail kolam, aktivasi dan deaktivasi kolam ke <i>webservice</i>	next sprint

Pada sprint ini story yang di pilih untuk di uraikan pada sprint kali ini adalah membuat fitur registrasi kolam dan aktivasi musim budidaya . Tujuan dari *sprint-2* ini adalah membuat halaman list kolam, registrasi kolam, detail kolam, aktivasi dan deaktivasi kolam dan mengintegrasikan halaman tersebut dengan *webservice*. Kendala yang dialami penulis pada sprint kali ini adalah banyaknya fitur yang harus di aplikasikan sehingga ada task yang harus dilanjukan pada sprint berikutnya. Berikut merupakan hasil dari peng�aan yang dilakukan selama *sprint 2*.

1. *Membuat Mock-up UI Halaman List kolam, Registrasi kolam, Detail kolam, Aktivasi dan Deaktivasi kolam*

Pembuatan konten dan fitur yang terdapat pada *mock-up UI* halaman list kolam, registrasi kolam, detail kolam, aktivasi dan deaktivasi kolam dilakukan

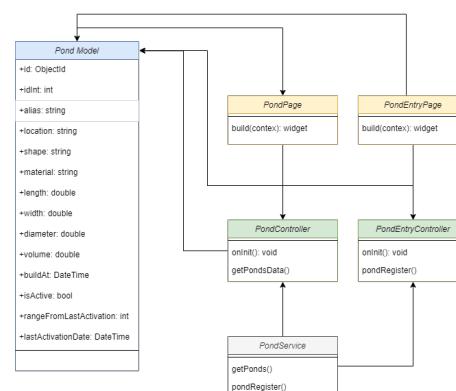
berdasarkan persetujuan product owner dan scrum master pada meeting sebelumnya. Mock-up UI dibuat menggunakan platform figma.



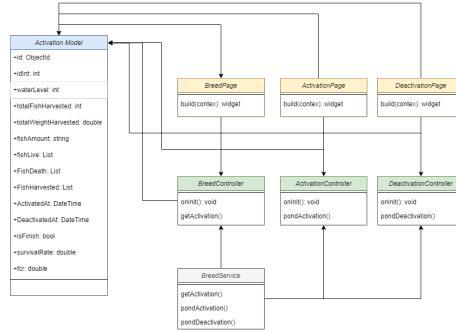
Gambar 4.7: Mock-up UI List kolam, Registrasi kolam, Detail kolam, Aktivasi dan Deaktivasi kolam

2. Class Diagram

Class Diagram menggambarkan kelas-kelas yang akan dipakai oleh sistem. Umumnya terdapat 3 kelas pada setiap module yaitu class model, controller, dan view. Pada sprint-2 penelitian kali ini penulis membuat 4 class yaitu model yang berwarna biru, view berwarna oranye, controller yang berwarna hijau, dan service yang berwarna kuning.



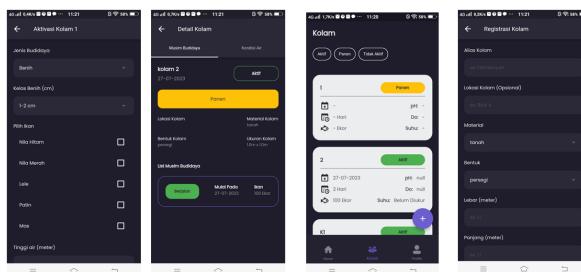
Gambar 4.8: Class Diagram Fitur Koam Sprint-2



Gambar 4.9: Class Diagram Fitur Aktivasi dan Deaktivasi Sprint-2

3. Menerapkan mockup-UI halaman list kolam, registrasi kolam, detail kolam, aktivasi dan deaktivasi kedalam code flutter

Setelah mock-up UI halaman list kolam, registrasi kolam, detail kolam, aktivasi dan deaktivasi dibuat, akan dilakukan pengimplementasian mock-up UI ke dalam aplikasi menggunakan flutter. Berikut adalah source code dari implementasi halaman list kolam, registrasi kolam, detail kolam, aktivasi dan deaktivasi yang terdapat pada lampiran 3 dan menghasilkan output halaman sebagai berikut



Gambar 4.10: Output dari code pada sprint 2

4. Analisis User Experience

pada halaman halaman list kolam terdapat card yang berisi informasi yang berguna bagi para pembudidaya untuk melakukan pemantauan terkait aktivitas yang terdapat pada masing-masing kolam. Sementara pada halaman registrasi kolam, pembudidaya harus memasukan data yang diperlukan untuk menambahkan kolam sesuai dengan kesepakatan saat meeting. Lalu pada halaman detail kolam terdapat data kolam terkait dan list musim budidaya yang telah berjalan di kolam tersebut. Sama halnya dengan aktivasi dan

deaktivasi pembudidaya harus memasukan data yang diperlukan sesuai dengan kesepakatan saat meeting.

5. Sprint 2 Review dan Sprint 3 Planning

Sprint 2 diakhiri dengan melakukan weekly meeting pada hari selasa dengan agenda melakukan review dan testing terkait hasil sprint 2 dan melakukan planning untuk sprint 3 dengan rincian:

(a) *Review dan Testing hasil dari sprint 2*

Telah dilakukan review dan testing oleh penulis selaku developer dengan Scrum Master. Setelah dilakukan testing, Scrum Master menyimpulkan bahwa penerapan fitur pada halaman list kolam, detail kolam, aktivasi kolam, deaktivasi kolam telah berjalan dengan baik namun belum di integrasikan dengan data dari API sehingga akan dilanjutkan kedalam sprint berikutnya.

(b) *Sprint Planning untuk Sprint 3*

Planning untuk sprint 3 yakni membuat fitur pemberian pakan pada aplikasi *Assistive Aquaculture Breeding Management*.

3. *Sprint 3*

Sprint-3 dilakukan sepekan pada tanggal 6 September 2022 sampai dengan 13 September 2022. Story ketiga pada *product backlog* yaitu membuat fitur pemberian pakan, ditambah dengan melanjutkan bagian yang belum selesai dari sprint sebelumnya yaitu mengintegrasikan halaman list kolam, registrasi kolam, aktivasi kolam, dan deaktivasi kolam dipecah menjadi beberapa *task* sebagai berikut.

Tabel 4.4: Sprint 3

No	Story	Task	Status
1	Melanjutkan bagian yang belum selesai dari sprint sebelumnya	Mengintegrasikan halaman list kolam, registrasi kolam, aktivasi kolam, dan deaktivasi kolam	selesai
2	Membuat fitur pemberian pakan	Membuat mockup-UI untuk fitur pemberian pakan	selesai

No	Story	Task	Status
3		Menerapkan <i>Mock-up UI</i> halaman pemberian pakan	selesai
4		Mengintegrasikan halaman pemberian pakan ke <i>webservice</i>	selesai

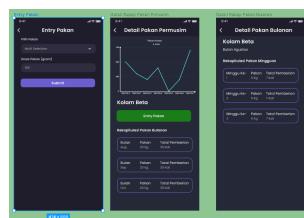
Pada *sprint-3* ini, story yang dipilih untuk diuraikan adalah membuat fitur pemberian pakan pada kolam budidaya. Tujuan dari *sprint-3* adalah membuat fitur pemberian pakan, serta mengintegrasikan halaman-halaman tersebut dengan *webservice*. Kendala yang dialami penulis pada *sprint-3* ini adalah banyaknya fitur yang harus diaplikasikan sehingga ada beberapa task yang harus dilanjutkan pada *sprint* berikutnya. Berikut merupakan hasil dari penggeraan yang dilakukan selama *sprint 3*.

1. *Mengintegrasikan halaman list kolam, registrasi kolam, aktivasi kolam, deaktivasi kolam dengan webservice*

sebelumnya data yang terdapat pada halaman list kolam, registrasi kolam, aktivasi kolam, deaktivasi kolam masih berupa data dummy, maka dari itu pada sprint ini penulis perlu mengintegrasikan halaman tersebut dengan web service, code tersebut dapat dilihat pada lampiran 4.

2. *Membuat Mock-up UI Halaman Pemberian pakan*

Pembuatan konten dan fitur yang terdapat pada *mock-up UI* halaman pemberian pakan dilakukan berdasarkan persetujuan product owner dan scrum master pada meeting sebelumnya. *Mock-up UI* dibuat menggunakan platform figma.



Gambar 4.11: *Mock-up UI pemberian pakan*

3. *Menerapkan mockup-UI halaman pemberian pakan, rekapitulasi pakan kedalam code flutter*

Setelah *mock-up UI pemberian pakan, rekapitulasi pakan* dibuat, akan dilakukan pengimplementasian *mock-up UI* ke dalam aplikasi menggunakan flutter. Berikut adalah source code dari implementasi halaman pemberian pakan, rekapitulasi pakan yang dikelompokan pada lampiran 4 dan menghasilkan output sebagai berikut.



Gambar 4.12: Output dari code pada rekapitulasi pakan

The screenshot shows a mobile application interface titled 'Entry Pakan Kolam 2'. It includes fields for 'Pilih Pakan' (selected: 'pelet'), 'Pilih Satuan' (selected: 'gram'), and 'Dosis Pakan' (input field containing 'ex: 2.1'). A large purple 'Submit' button is located at the bottom of the form.

Gambar 4.13: Output dari code untuk halaman entry pakan

4. Mengintegrasikan fitur pemberian pakan dengan webservice

Sebelumnya, setiap data pada fitur masih berupa data dummy sehingga perlu diintegrasikan dengan webservice agar aplikasi dapat berjalan dengan data yang asli. Hal yang dilakukan dalam mengintegrasikan fitur pemberian pakan dengan webservice terdapat pada lampiran 4.

5. Analisis *User Experience*

Pada halaman pemberian pakan, pembudidaya harus memasukan dosis pakan dan jenis pakan yang diberikan sehingga data tersebut dapat diolah untuk output FCR nanti saat panen. Di halaman rekapitulasi pakan, terdapat grafik mengenai statistik jumlah pakan yang telah diberikan pada musim budidaya yang berjalan sehingga pembudidaya dapat dengan mudah mengetahui statistik rekap pakan dari setiap musim budidaya yang berjalan, selain itu terdapat juga list mengenai pakan yang telah diberikan yang berisi informasi dosis dan waktu pembudidaya saat memberikan pakan.

6. Sprint 3 Review dan Sprint 4 Planning

Sprint 3 diakhiri dengan melakukan weekly meeting pada hari selasa dengan agenda melakukan review dan testing terkait hasil sprint 3 dan melakukan planning untuk sprint 4 dengan rincian:

(a) *Review dan Testing hasil dari sprint 3*

Telah dilakukan review dan testing oleh penulis selaku developer dengan Scrum Master. Setelah dilakukan testing, Scrum Master menyimpulkan bahwa penerapan fitur pemberian pakan telah berjalan dengan baik.

Tabel 4.5: Unit testing Halaman Daftar Kolam.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Saat ikon (+) ditekan maka akan menampilkan halaman registrasi kolam	✓		Diterima
Saat card kolam ditekan akan menampilkan halaman detail kolam	✓		Diterima

Tabel 4.6: Unit testing Halaman Registrasi Kolam.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
ketika mengisi form registrasi kolam dengan data yang sesuai dan menekan submit, maka kolam baru akan ditambahkan	✓		Diterima

Tabel 4.7: Unit testing Halaman Detail Kolam.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
ketika tombol kualitas air ditekan maka akan menampilkan halaman kualitas air	✓		Diterima
ketikan menekan salah satu list dari masa budidaya akan menampilkan halaman detail masa budidaya	✓		Diterima
ketika status kolam tidak aktif atau panen dan menekan tombol start budidaya, maka akan menampilkan halaman aktivasi kolam	✓		Diterima
ketika status kolam aktif dan menekan tombol aktif, maka akan menampilkan halaman deaktivasi kolam	✓		Diterima

Tabel 4.8: Unit testing Halaman Registrasi Kolam.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
ketika mengisi form registrasi kolam dengan data yang sesuai dan menekan submit, maka kolam baru akan ditambahkan	✓		Diterima

Tabel 4.9: Unit testing Halaman Aktivasi Kolam.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
ketika mengisi form aktivasi kolam dengan data yang sesuai dan menekan submit, maka musim budidaya baru akan ditambahkan dan status kolam menjadi aktif	✓		Diterima

Tabel 4.10: Unit testing Halaman Deaktivasi Kolam.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
ketika mengisi form deaktivasi kolam dengan data yang sesuai dan menekan submit, maka musim budidaya yang barjalan akan berhenti status kolam menjadi tidak aktif	✓		Diterima

Tabel 4.11: Unit testing Halaman Detail Masa Budidaya.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Ketika tombol rekapitulasi pakan ditekan, maka akan ditampilkan halaman rekapitulasi pakan	✓		Diterima
Ketika tombol rekapitulasi kematian ikan ditekan, maka akan ditampilkan halaman rekapitulasi kematian ikan	✓		Diterima
Ketika tombol rekapitulasi grading ikan ditekan, maka akan ditampilkan halaman rekapitulasi grading ikan	✓		Diterima
Ketika tombol treatment ditekan, maka akan ditampilkan halaman treatment kolam	✓		Diterima
Ketika tombol sortir ditekan, maka akan ditampilkan halaman sortir kolam	✓		Diterima

Tabel 4.12: Unit testing Halaman Rekapitulasi Pakan.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Ketika menekan list data rekapitulasi pakan, maka akan ditampilkan detail rekapitulasi pakan	✓		Diterima
Ketika menekan list data rekapitulasi pakan, maka akan ditampilkan detail rekapitulasi pakan	✓		Diterima
Saat tombol entry pakan ditekan maka akan menampilkan halaman entry pakan	✓		Diterima
ketika mengisi form rekapitulasi pakan dengan data yang sesuai dan menekan submit, data rekapitulasi pakan akan ditambahkan	✓		Diterima

(b) *Sprint Planning untuk Sprint 4*

Planning untuk sprint 4 yakni membuat fitur grading ikan pada aplikasi *Assistive Aquaculture Breeding Management*.

4. Sprint 4

Sprint-4 dilakukan sepekan pada tanggal 13 September 2022 sampai dengan 20 september 2022. Story keempat pada *product backlog* yaitu membuat grading berat dipecah menjadi beberapa *task* sebagai berikut.

Tabel 4.13: *Sprint 4*

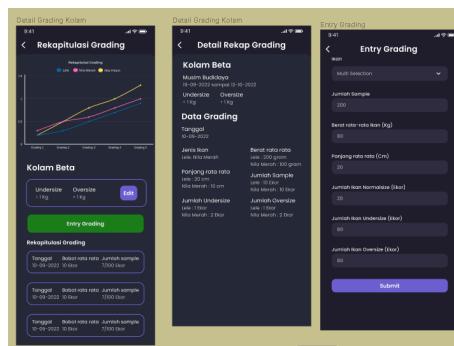
No	Story	Task	Status
1	Membuat fitur grading berat ikan	Membuat <i>Mock-up UI</i> halaman rekapitulasi grading, detail grading, entry grading	selesai
2		Menerapkan <i>Mock-up UI</i> halaman rekapitulasi grading, detail grading, entry grading ke Flutter	selesai

No	Story	Task	Status
3		Mengintegrasikan halaman rekapitulasi grading, detail grading, entry grading ke webservice	selesai

Pada sprint keempat ini story yang di pilih untuk di uraikan pada sprint kali ini adalah membuat halaman rekapitulasi grading, detail grading, entry grading. Tujuan dari *sprint-4* ini adalah membuat fitur grading berat ikan dan mengintegrasikan halaman tersebut dengan webservice yang sudah dibuat oleh penelitian Andri Rahmanto.

1. Membuat Mock-up UI Fitur Grading

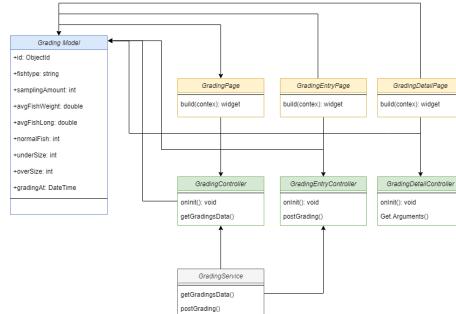
Pembuatan konten dan fitur yang terdapat pada *mock-up UI* fitur grading dilakukan berdasarkan persetujuan product owner dan scrum master pada meeting sebelumnya. Mock-up UI dibuat menggunakan platform figma.



Gambar 4.14: *Mock-up UI Fitur Grading*

2. Class Diagram

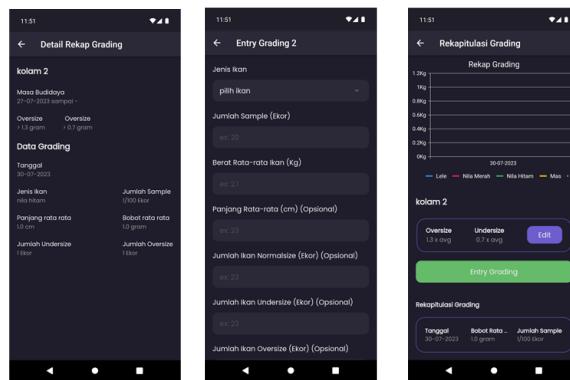
Class Diagram menggambarkan kelas-kelas yang akan dipakai oleh sistem. Umumnya terdapat 3 kelas pada setiap module yaitu class model, controller, dan view. Pada sprint-4 penelitian kali ini penulis membuat 4 class yaitu model yang berwarna biru, view berwarna oranye, controller yang berwarna hijau, dan service yang berwarna kuning.



Gambar 4.15: Class Diagram Fitur Fitur Sprint-4

3. Menerapkan Mockup-UI Fitur Grading kedalam code flutter

Setelah *mock-up UI* fitur grading hingga prototype dibuat, akan dilakukan pengimplementasian *mock-up UI* ke dalam aplikasi menggunakan flutter. Source code dari implementasi fitur grading terdapat pada lampiran 5 yang menghasilkan output halaman sebagai berikut



Gambar 4.16: Output dari code pada sprint 4

4. Mengintegrasikan fitur grading dengan webservice

Sebelumnya, setiap data pada fitur masih berupa data dummy sehingga perlu diintegrasikan dengan webservice agar aplikasi dapat berjalan dengan data yang asli. Hal yang dilakukan dalam mengintegrasikan fitur grading dengan webservice terdapat pada lampiran 5.

5. Analisis User Experience

Pada halaman grading, pembudidaya harus jenis ikan dan data klasifikasi ukuran ikan. Di halaman rekapitulasi grading, terdapat grafik mengenai statistik pertumbuhan ikan pada musim budidaya yang berjalan sehingga

pembudidaya dapat dengan mudah mengetahui perkembangan ukuran ikan dari waktu ke waktu, selain itu terdapat juga list mengenai grading yang telah dilakukan yang berisi informasi yang berhubungan dengan ukuran ikan saat dilakukan grading.

6. Sprint 4 Review dan Sprint 5 Planning

Sprint 4 diakhiri dengan melakukan weekly meeting pada hari selasa dengan agenda melakukan review dan testing terkait hasil sprint 4 dan melakukan planning untuk sprint 5 dengan rincian:

(a) *Review dan Testing hasil dari sprint 4*

Telah dilakukan review dan testing oleh penulis selaku developer dengan Scrum Master. Setelah dilakukan testing, Scrum Master menyimpulkan bahwa fitur grading berat ikan telah berjalan dengan baik.

Tabel 4.14: Unit testing Halaman Rekapitulasi Grading.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Ketika menekan list data rekapitulasi grading, maka akan ditampilkan detail rekapitulasi grading	✓		Diterima
Ketika menekan list data rekapitulasi grading, maka akan ditampilkan detail rekapitulasi grading	✓		Diterima
Saat tombol entry grading ditekan maka akan menampilkan halaman entry grading	✓		Diterima
ketika mengisi form rekapitulasi grading dengan data yang sesuai dan menekan submit, data rekapitulasi grading akan ditambahkan	✓		Diterima

(b) *Sprint Planning untuk Sprint 5*

Planning untuk sprint 5 yakni membuat fitur rekapitulasi kematian ikan pada aplikasi *Assistive Aquaculture Breeding Management*.

5. Sprint 5

Sprint-5 dilakukan sepekan pada tanggal 20 September 2022 sampai dengan 27 september 2022. *Story* kelima pada *product backlog* yaitu membuat fitur rekapitulasi kematian ikan dipecah menjadi beberapa *task* sebagai berikut.

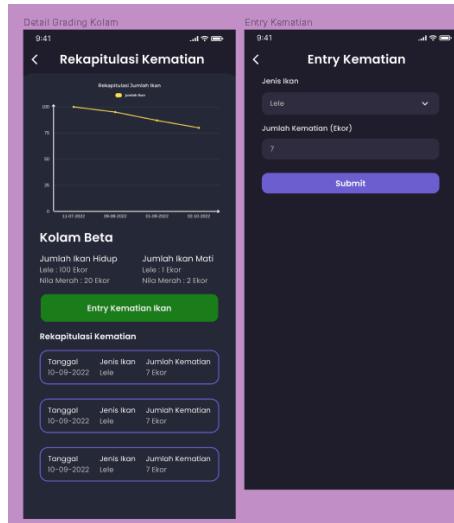
Tabel 4.15: Sprint 5

No	Story	Task	Status
1	Membuat fitur rekapitulasi kematian ikan	Membuat <i>Mock-up UI</i> halaman rekapitulasi kematian ikan, entry kematian ikan	selesai
2		Menerapkan <i>Mock-up UI</i> halaman rekapitulasi kematian ikan, entry kematian ikan ke Flutter	selesai
3		Mengintegrasikan halaman rekapitulasi kematian ikan, entry kematian ikan ke <i>webservice</i>	selesai

Pada sprint kelima ini story yang di pilih untuk di uraikan pada sprint kali ini adalah membuat halaman rekapitulasi kematian ikan, entry kematian ikan. Tujuan dari *sprint-5* ini adalah membuat fitur rekapitulasi kematian ikan dan mengintegrasikan halaman tersebut dengan *webservice* yang sudah dibuat oleh penelitian Andri Rahmanto.

1. Membuat *Mock-up UI* Fitur Rekapitulasi Kematian Ikan

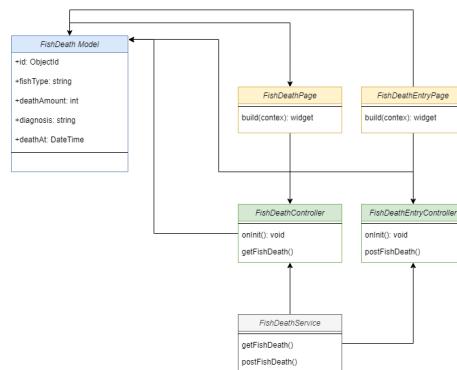
Pembuatan konten dan fitur yang terdapat pada *mock-up UI* fitur rekapitulasi kematian ikan dilakukan berdasarkan persetujuan product owner dan scrum master pada meeting sebelumnya. *Mock-up UI* dibuat menggunakan platform figma.



Gambar 4.17: Mock-up UI Fitur Rekapitulasi Kematian Ikan

2. Class Diagram

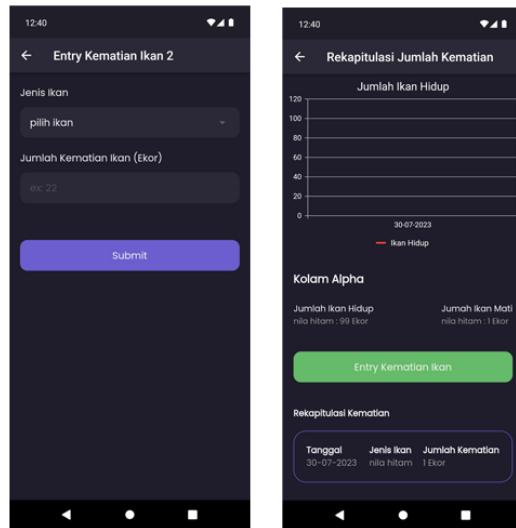
Class Diagram menggambarkan kelas-kelas yang akan dipakai oleh sistem. Umumnya terdapat 3 kelas pada setiap module yaitu class model, controller, dan view. Pada sprint-5 penelitian kali ini penulis membuat 4 class yaitu model yang berwarna biru, view berwarna oranye, controller yang berwarna hijau, dan service yang berwarna kuning.



Gambar 4.18: Class Diagram Fitur Sprint-5

3. Menerapkan Mockup-UI fitur rekapitulasi kematian ikan kedalam code flutter

Setelah itu, akan dilakukan pengimplementasian mock-up UI ke dalam aplikasi menggunakan flutter. Pada lampiran 6 source code dari implementasi fitur grading yang dikelompokan berdasarkan halaman yang menghasilkan output halaman seperti dibawah ini.



Gambar 4.19: Output dari code pada sprint 5

4. Mengintegrasikan fitur rekapitulasi kematian dengan webservice

Sebelumnya, setiap data pada fitur masih berupa data dummy sehingga perlu diintegrasikan dengan webservice agar aplikasi dapat berjalan dengan data yang asli. Hal yang dilakukan dalam mengintegrasikan fitur rekapitulasi kematian ikan dengan webservice terdapat pada lampiran 6.

5. Analisis User Experience

Pada halaman entry rekapitulasi kematian, pembudidaya harus jenis ikan dan data jumlah kematian. Di halaman rekapitulasi kematian, terdapat grafik mengenai statistik jumlah ikan pada musim budidaya yang berjalan sehingga pembudidaya dapat dengan mudah mengetahui penurunan jumlah ikan akibat kematian. Selain itu terdapat juga list mengenai data kematian yang telah dimasukan yang berisi informasi yang berhubungan dengan kematian ikan.

6. Sprint 5 Review dan Sprint 6 Planning

Sprint 5 diakhiri dengan melakukan weekly meeting pada hari selasa dengan agenda melakukan review dan testing terkait hasil sprint 5 dan melakukan planning untuk sprint 6 dengan rincian:

(a) *Review dan Testing hasil dari sprint 5*

Telah dilakukan review dan testing oleh penulis selaku developer dengan Scrum Master. Setelah dilakukan testing, Scrum Master menyimpulkan bahwa fitur rekapitulasi kematian ikan telah berjalan dengan baik.

Tabel 4.16: Unit testing Halaman Rekapitulasi Kematian.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Ketika menekan list data rekapitulasi pakan, maka akan ditampilkan detail rekapitulasi pakan	✓		Diterima
Saat tombol entry kematian ditekan maka akan menampilkan halaman entry kematian	✓		Diterima
ketika mengisi form rekapitulasi kematian dengan data yang sesuai dan menekan submit, data rekapitulasi kematian akan ditambahkan	✓		Diterima

(b) *Sprint Planning untuk Sprint 6*

Planning untuk sprint 6 yakni membuat fitur treatment kolam pada aplikasi *Assistive Aquaculture Breeding Management*.

6. Sprint 6

Sprint-6 dilakukan sepekan pada tanggal 27 September 2022 sampai dengan 4 oktober 2022. Story keenam pada *product backlog* yaitu membuat fitur treatment kolam dipecah menjadi beberapa *task* sebagai berikut.

Tabel 4.17: *Sprint 6*

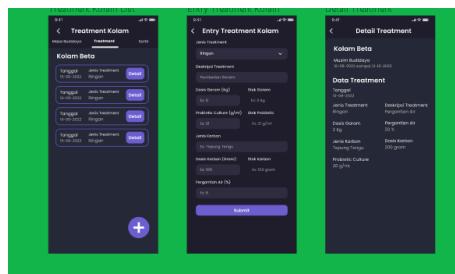
No	Story	Task	Status
1	Membuat fitur treatment kolam	Membuat <i>Mock-up UI</i> halaman list treatment, detail treatment, entry treatment	selesai
2		Menerapkan <i>Mock-up UI</i> halaman list treatment, detail treatment, entry treatment ke Flutter	selesai

Pada sprint keenam ini story yang di pilih untuk di uraikan pada sprint kali ini adalah membuat halaman rekapitulasi treatment, detail treatment, entry treatment. Tujuan dari *sprint-6* ini adalah membuat fitur treatment berat ikan dan

mengintegrasikan halaman tersebut dengan webservice yang sudah dibuat oleh penelitian Andri Rahmanto.

1. Membuat Mock-up UI Fitur Treatment Kolam

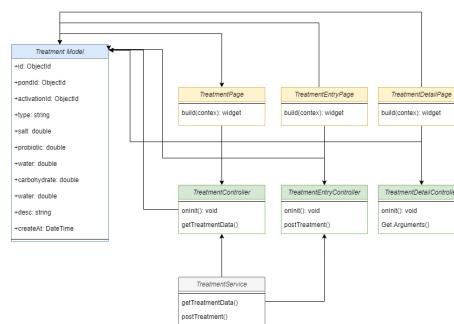
Pembuatan konten dan fitur yang terdapat pada *mock-up UI* fitur treatment kolam dilakukan berdasarkan persetujuan product owner dan scrum master pada meeting sebelumnya. *Mock-up UI* dibuat menggunakan platform figma.



Gambar 4.20: *Mock-up UI Fitur treatment*

2. Class Diagram

Class Diagram menggambarkan kelas-kelas yang akan dipakai oleh sistem. Umumnya terdapat 3 kelas pada setiap module yaitu class model, controller, dan view. Pada sprint-6 penelitian kali ini penulis membuat 4 class yaitu model yang berwarna biru, view berwarna oranye, controller yang berwarna hijau, dan service yang berwarna kuning.

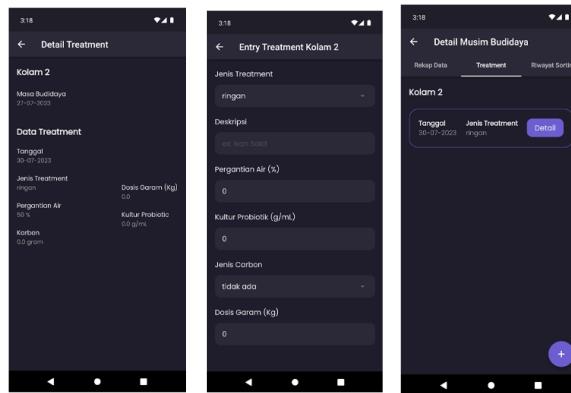


Gambar 4.21: *Class Diagram Fitur Sprint-6*

3. Menerapkan Mockup-UI Fitur Treatment Kolam kedalam code flutter

Setelah *mock-up UI* fitur treatment kolam, akan dilakukan pengimplementasian *mock-up UI* ke dalam aplikasi menggunakan flutter. Pada

lampiran 7 terdapat source code dari implementasi fitur treatment yang dikelompokan berdasarkan halaman yang menghasilkan halaman seperti dibawah ini.



Gambar 4.22: Output dari code pada sprint 6

4. Analisis *User Experience*

Pada halaman entry treatment, pembudidaya harus memasukan data yang diperlukan untuk melakukan treatment kolam sesuai dengan kesepakatan saat meeting. Selain itu terdapat juga list mengenai data treatment yang telah dimasukan yang berisi informasi yang berhubungan dengan treatment kolam. Terdapat pula halaman detail treatment yang berisi informasi yang lebih detail terkait treatment yang telah dilakukan.

5. Sprint 6 Review dan Sprint 7 Planning

Sprint 6 diakhiri dengan melakukan weekly meeting pada hari selasa dengan agenda melakukan review dan testing terkait hasil sprint 6 dan melakukan planning untuk sprint 7 dengan rincian:

(a) *Review dan Testing hasil dari sprint 6*

Telah dilakukan review dan testing oleh penulis selaku developer dengan Scrum Master. Setelah dilakukan testing, Scrum Master menyimpulkan bahwa fitur treatment kolam telah berjalan dengan baik.

(b) *Sprint Planning untuk Sprint 7*

Planning untuk sprint 7 yakni membuat halaman untuk fitur treatment kolam pada aplikasi *Assistive Aquaculture Breeding Management*.

7. Sprint 8

Sprint-8 dilakukan sepekan pada tanggal 4 oktober 2022 sampai dengan 11 oktober 2022. *Story* ketujuh pada *product backlog* yaitu mengintegrasikan fitur treatment kolam dipecah menjadi beberapa *task* sebagai berikut.

Tabel 4.18: *Sprint 7*

No	Story	Task	Status
1	Mengintegrasikan fitur treatment kolam dengan webservice	Mengintegrasikan fitur treatment kolam dengan webservice	selesai

Pada sprint ketujuh ini story yang di pilih untuk di uraikan pada sprint kali ini adalah membuat halaman rekapitulasi treatment kolam, entry treatment kolam. Tujuan dari *sprint-7* ini adalah membuat fitur rekapitulasi treatment kolam dan mengintegrasikan halaman tersebut dengan webservice yang sudah dibuat oleh penelitian Andri Rahmanto.

1. Mengintegrasikan fitur treatment kolam dengan webservice

Sebelumnya pada sprint keenam, setiap data pada fitur treatment masih berupa data dummy sehingga perlu diintegrasikan dengan webservice agar aplikasi dapat berjalan dengan data yang asli. Hal yang dilakukan dalam mengintegrasikan fitur treatment dengan webservice terdapat pada lampiran 8.

2. Sprint 7 Review dan Sprint 8 Planning

Sprint 7 diakhiri dengan melakukan weekly meeting pada hari selasa dengan agenda melakukan review dan testing terkait hasil sprint 7 dan melakukan planning untuk sprint 8 dengan rincian:

(a) Review dan Testing hasil dari sprint 8

Telah dilakukan review dan testing oleh penulis selaku developer dengan Scrum Master. Setelah dilakukan testing, Scrum Master menyimpulkan bahwa fitur treatment kolam telah berjalan dengan baik.

Tabel 4.19: Unit testing Halaman Rekapitulasi Treatment.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Ketika menekan list data treatment, maka akan ditampilkan detail treatment	✓		Diterima
Saat ikon (+) ditekan maka akan menampilkan halaman entry treatment	✓		Diterima
ketika mengisi form treatment dengan data yang sesuai dan menekan submit, data treatment akan ditambahkan	✓		Diterima

(b) *Sprint Planning untuk Sprint 8*

Planning untuk sprint 8 yakni membuat fitur pencatatan kualitas air harian pada aplikasi *Assistive Aquaculture Breeding Management*.

8. Sprint 8

Sprint-8 dilakukan sepekan pada tanggal 11 oktober 2022 sampai dengan 18 oktober 2022. *Story* kedelapan pada *product backlog* yaitu membuat fitur pencatatan kualitas air harian dipecah menjadi beberapa *task* sebagai berikut.

Tabel 4.20: *Sprint 8*

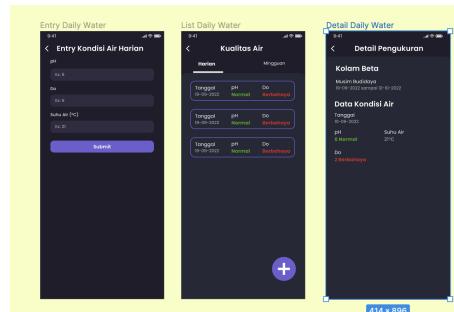
No	Story	Task	Status
1	Membuat fitur pencatatan kualitas air harian	Membuat halaman list pencatatan kualitas air harian, entry kualitas air harian, detail kualitas air harian	selesai
2		Menerapkan Mock-up UI halaman list pencatatan kualitas air harian, entry kualitas air harian, detail kualitas air harian ke Flutter	selesai

No	Story	Task	Status
3		Mengintegrasikan halaman pencatatan kualitas air harian, entry kualitas air harian, detail kualitas air harian ke <i>webservice</i>	selesai

Pada sprint kedelapan ini story yang di pilih untuk di uraikan pada sprint kali ini adalah membuat halaman pencatatan kualitas air harian, entry kualitas air harian, detail kualitas air harian. Tujuan dari *sprint-8* ini adalah membuat fitur pencatatan kualitas air harian dan mengintegrasikan halaman tersebut dengan webservice yang sudah dibuat oleh penelitian Andri Rahmanto.

1. Membuat Mock-up UI Fitur Pencatatan Kualitas Air Harian

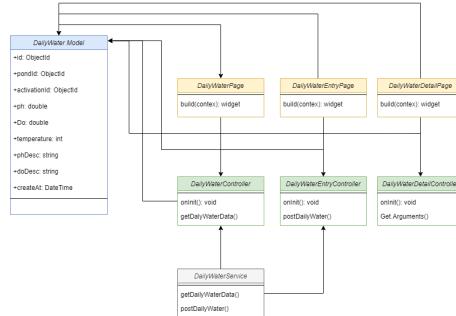
Pembuatan konten dan fitur yang terdapat pada *mock-up UI* fitur pencatatan kualitas air harian dilakukan berdasarkan persetujuan product owner dan scrum master pada meeting sebelumnya. Mock-up UI dibuat menggunakan platform figma.



Gambar 4.23: *Mock-up UI Fitur Pencatatan Kualitas Air Harian*

2. Class Diagram

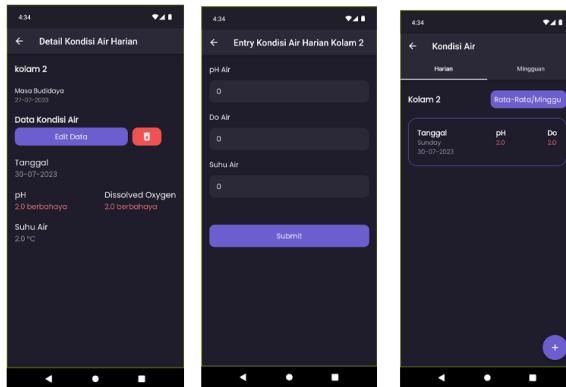
Class Diagram menggambarkan kelas-kelas yang akan dipakai oleh sistem. Umumnya terdapat 3 kelas pada setiap module yaitu class model, controller, dan view. Pada sprint-8 penelitian kali ini penulis membuat 4 class yaitu model yang berwarna biru, view berwarna oranye, controller yang berwarna hijau, dan service yang berwarna kuning.



Gambar 4.24: Class Diagram Fitur Sprint-8

3. Menerapkan Mockup-UI Fitur Pencatatan Kualitas Air Harian kedalam code flutter

Setelah itu, akan dilakukan pengimplementasian *mock-up UI* ke dalam aplikasi menggunakan flutter. Pada lampiran 9 terdapat source code dari implementasi fitur pencatatan kualitas air harian yang dikelompokan berdasarkan halaman yang menghasilkan output halaman sebagai seperti dibawah ini.



Gambar 4.25: Output dari code pada sprint 8

4. Mengintegrasikan fitur pencatatan kualitas air harian dengan webservice

Sebelumnya, setiap data pada fitur masih berupa data dummy sehingga perlu diintegrasikan dengan webservice agar aplikasi dapat berjalan dengan data yang asli. Hal yang dilakukan dalam mengintegrasikan fitur pencatatan kualitas air harian dengan webservice terdapat pada lampiran 9.

5. Analisis User Experience

Pada halaman entry daily water quality, pembudidaya harus memasukan

data yang diperlukan untuk melakukan daily water quality sesuai dengan kesepakatan saat meeting, seperti pH, Dissolved Oxygen, dan suhu. Selain itu terdapat juga list mengenai data daily water quality yang telah dimasukan yang berisi informasi yang berhubungan dengan daily water quality. Terdapat pula halaman detail daily water quality yang berisi informasi yang lebih detail terkait daily water quality yang telah dilakukan.

6. Sprint 8 Review dan Sprint 9 Planning

Sprint 8 diakhiri dengan melakukan weekly meeting pada hari selasa dengan agenda melakukan review dan testing terkait hasil sprint 8 dan melakukan planning untuk sprint 9 dengan rincian:

(a) *Review dan Testing hasil dari sprint 8*

Telah dilakukan review dan testing oleh penulis selaku developer dengan Scrum Master. Setelah dilakukan testing, Scrum Master menyimpulkan bahwa fitur pencatatan kualitas air harian telah berjalan dengan baik.

(b) *Sprint Planning untuk Sprint 9*

Planning untuk sprint 9 yakni membuat fitur pencatatan kualitas air mingguan pada aplikasi *Assistive Aquaculture Breeding Management*.

9. *Sprint 9*

Sprint-9 dilakukan sepekan pada tanggal 18 oktober 2022 sampai dengan 25 oktober 2022. Story kesembilan pada *product backlog* yaitu membuat fitur pencatatan kualitas air mingguan dipecah menjadi beberapa *task* sebagai berikut.

Tabel 4.21: Sprint 9

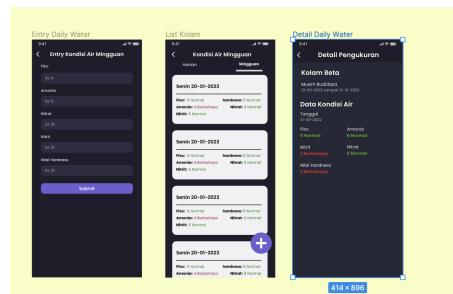
No	Story	Task	Status
1	Membuat fitur pencatatan kualitas air mingguan	Membuat halaman list pencatatan kualitas air mingguan, entry kualitas air mingguan, detail kualitas air mingguan	selesai

No	Story	Task	Status
2		Menerapkan <i>Mock-up UI</i> halaman list pencatatan kualitas air mingguan, entry kualitas air mingguan, detail kualitas air mingguan ke Flutter	selesai
3		Mengintegrasikan halaman pencatatan kualitas air mingguan, entry kualitas air mingguan, detail kualitas air mingguan ke <i>webservice</i>	selesai

Pada sprint kesembilan ini story yang di pilih untuk di uraikan pada sprint kali ini adalah membuat halaman pencatatan kualitas air mingguan, entry kualitas air mingguan, detail kualitas air mingguan. Tujuan dari *sprint-9* ini adalah membuat fitur pencatatan kualitas air mingguan dan mengintegrasikan halaman tersebut dengan *webservice* yang sudah dibuat oleh penelitian Andri Rahmanto.

1. Membuat *Mock-up UI* Fitur Pencatatan Kualitas air mingguan

Pembuatan konten dan fitur yang terdapat pada *mock-up UI* fitur pencatatan kualitas air mingguan dilakukan berdasarkan persetujuan product owner dan scrum master pada meeting sebelumnya. *Mock-up UI* dibuat menggunakan platform figma.

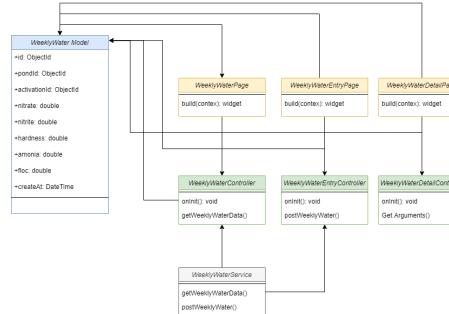


Gambar 4.26: *Mock-up UI* Fitur Pencatatan Kualitas air mingguan

2. Class Diagram

Class Diagram menggambarkan kelas-kelas yang akan dipakai oleh sistem. Umumnya terdapat 3 kelas pada setiap module yaitu class model,

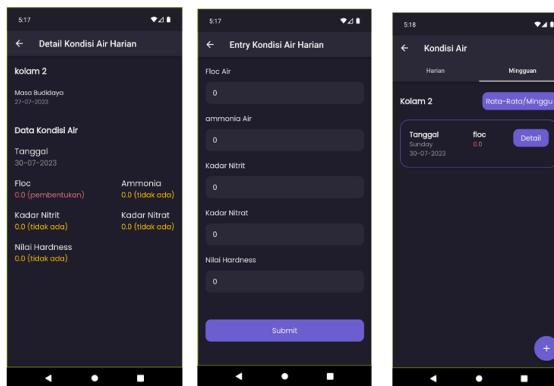
controller, dan view. Pada sprint-9 penelitian kali ini penulis membuat 4 class yaitu model yang berwarna biru, view berwarna oranye, controller yang berwarna hijau, dan service yang berwarna kuning.



Gambar 4.27: Class Diagram Fitur Sprint-9

3. Menerapkan Mockup-UI Fitur Pencatatan Kualitas air mingguan kedalam code flutter

Setelah itu, akan dilakukan pengimplementasian *mock-up UI* ke dalam aplikasi menggunakan flutter. Pada lampiran 10 terdapat source code dari implementasi fitur pencatatan kualitas air mingguan yang dikelompokan berdasarkan halaman yang menghasilkan output halaman seperti dibawah ini



Gambar 4.28: Output dari code pada sprint 9

4. Mengintegrasikan fitur pencatatan kualitas air mingguan dengan webservice

Sebelumnya, setiap data pada fitur masih berupa data dummy sehingga perlu diintegrasikan dengan webservice agar aplikasi dapat berjalan dengan data yang asli. Hal yang dilakukan dalam mengintegrasikan fitur pencatatan kualitas air mingguan dengan webservice terdapat pada lampiran 10.

5. Analisis *User Experience*

Pada halaman entry weekly water quality, pembudidaya harus memasukan data yang diperlukan untuk melakukan weekly water quality sesuai dengan kesepakatan saat meeting, seperti kadar nitrit, nitrat, amonia, dan nilai kekerasan air. Selain itu terdapat juga list mengenai data weekly water quality yang telah dimasukan yang berisi informasi yang berhubungan dengan weekly water quality. Terdapat pula halaman detail weekly water quality yang berisi informasi yang lebih detail terkait weekly water quality yang telah dilakukan.

6. Sprint 9 Review dan Sprint 10 Planning

Sprint 9 diakhiri dengan melakukan weekly meeting pada hari selasa dengan agenda melakukan review dan testing terkait hasil sprint 9 dan melakukan planning untuk sprint 10 dengan rincian:

(a) *Review dan Testing hasil dari sprint 9*

Telah dilakukan review dan testing oleh penulis selaku developer dengan Scrum Master. Setelah dilakukan testing, Scrum Master menyimpulkan bahwa fitur pencatatan kualitas air mingguan telah berjalan dengan baik.

Tabel 4.22: Unit testing Halaman Kualitas Air.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Ketika memilih musim budidaya dan maka akan ditampilkan list pengontrolan kualitas air kolam	✓		Diterima
Ketika menekan list data pengontrolan kualitas air, maka akan ditampilkan detail pengontrolan kualitas air kolam	✓		Diterima
Saat ikon (+) ditekan maka akan menampilkan halaman entry pengontrolan kualitas air kolam	✓		Diterima
ketika mengisi form pengontrolan kualitas air kolam dengan data yang sesuai dan menekan submit, data pengontrolan kualitas air akan ditambahkan	✓		Diterima

(b) *Sprint Planning untuk Sprint 10*

Planning untuk sprint 10 yakni membuat fitur sortir kolam mingguan pada aplikasi *Assistive Aquaculture Breeding Management*.

10. Sprint 10

Sprint-10 dilakukan sepekan pada tanggal 25 oktober 2022 sampai dengan 1 november 2022. Story kesepuluh pada *product backlog* yaitu membuat fitur perpindahan antar kolam dipecah menjadi beberapa *task* sebagai berikut.

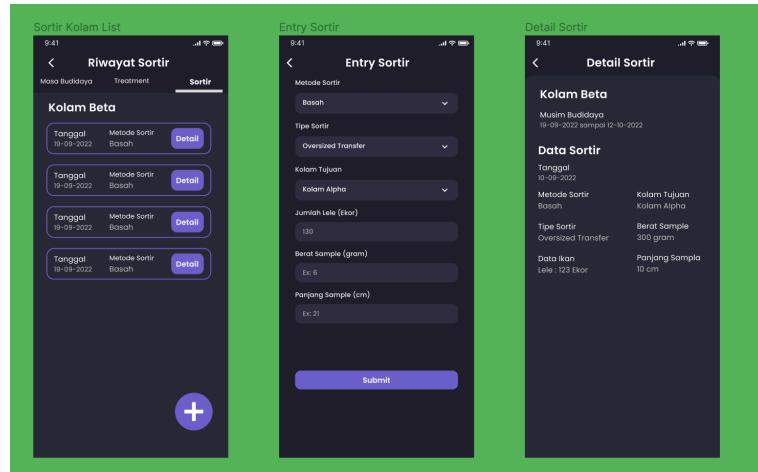
Tabel 4.23: Sprint 10

No	Story	Task	Status
1	Membuat fitur sortir kolam	Membuat <i>Mock-up UI</i> halaman list sortir, detail sortir, entry sortir	selesai
2		Menerapkan <i>Mock-up UI</i> halaman list sortir, detail sortir, entry sortir ke Flutter	selesai
3		Mengintegrasikan halaman list soritr, entry soritr, detail soritr ke <i>webservice</i>	selesai

Pada sprint kesepuluh ini story yang di pilih untuk di uraikan pada sprint kali ini adalah membuat fitur sortir. Tujuan dari *sprint-10* ini adalah membuat fitur sortir ikan dan mengintegrasikan halaman tersebut dengan *webservice* yang sudah dibuat oleh penelitian Andri Rahmanto.

1. Membuat *Mock-up UI* Fitur Sortir Kolam

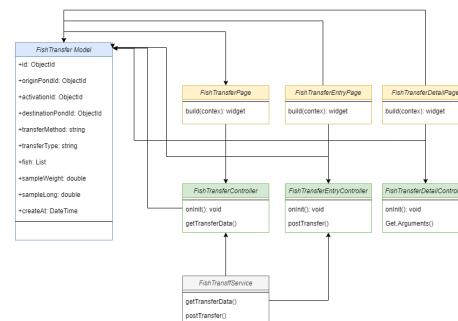
Pembuatan konten dan fitur yang terdapat pada *mock-up UI* fitur sortir kolam dilakukan berdasarkan persetujuan product owner dan scrum master pada meeting sebelumnya. *Mock-up UI* dibuat menggunakan platform figma.



Gambar 4.29: Mock-up UI Fitur Sortir

2. Class Diagram

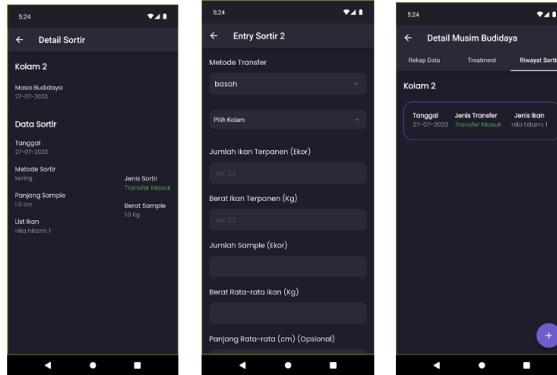
Class Diagram menggambarkan kelas-kelas yang akan dipakai oleh sistem. Umumnya terdapat 3 kelas pada setiap module yaitu class model, controller, dan view. Pada sprint-10 penelitian kali ini penulis membuat 4 class yaitu model yang berwarna biru, view berwarna oranye, controller yang berwarna hijau, dan service yang berwarna kuning.



Gambar 4.30: Class Diagram Fitur Sprint-10

3. Menerapkan Mockup-UI Fitur Sortir Kolam kedalam code flutter

Setelah mock-up UI fitur Sortir kolam, akan dilakukan pengimplementasian mock-up UI ke dalam aplikasi menggunakan flutter. Pada lampiran 11 terdapat source code dari implementasi fitur sortir yang dikelompokan berdasarkan halaman yang menghasilkan output halaman seperti dibawah ini.



Gambar 4.31: Output dari code pada sprint 10

4. Mengintegrasikan fitur pencatatan kualitas air mingguan dengan webservice

Sebelumnya, setiap data pada fitur masih berupa data dummy sehingga perlu diintegrasikan dengan webservice agar aplikasi dapat berjalan dengan data yang asli. Hal yang dilakukan dalam mengintegrasikan fitur sortir ikan dengan webservice terdapat pada lampiran 11.

5. Analisis User Experience

Pada halaman entry sortir kolam/transfer ikan, pembudidaya harus memasukan data yang diperlukan untuk melakukan sortir kolam/transfer ikan sesuai dengan kesepakatan saat meeting. Selain itu terdapat juga list mengenai data sortir kolam/transfer ikan yang telah dimasukan yang berisi informasi yang berhubungan dengan sortir kolam/transfer ikan. Terdapat pula halaman detail sortir kolam/transfer ikan yang berisi informasi yang lebih detail terkait sortir kolam/transfer ikan yang telah dilakukan.

6. Sprint 10 Review dan Sprint 11 Planning

Sprint 10 diakhiri dengan melakukan weekly meeting pada hari selasa dengan agenda melakukan review dan testing terkait hasil sprint 10 dan melakukan planning untuk sprint 11 dengan rincian:

(a) *Review dan Testing hasil dari sprint 10*

Telah dilakukan review dan testing oleh penulis selaku developer dengan Scrum Master. Setelah dilakukan testing, Scrum Master menyimpulkan bahwa fitur transfer/sortir ikam telah berjalan dengan baik.

Tabel 4.24: Unit testing Halaman Rekapitulasi Sortir.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Ketika menekan list data sortir, maka akan ditampilkan detail sortir	✓		Diterima
Saat ikon (+) ditekan maka akan menampilkan halaman entry sortir	✓		Diterima
ketika mengisi form sortir dengan data yang sesuai dan menekan submit, data sortir akan ditambahkan	✓		Diterima

(b) *Sprint Planning untuk Sprint 11*

Planning untuk sprint 11 yakni membuat fitur multi user pada aplikasi *Assistive Aquaculture Breeding Management*.

11. *Sprint 11*

Sprint-11 dilakukan sepekan pada tanggal 1 november 2022 sampai dengan 8 november 2022. *Story* kesebelas pada *product backlog* yaitu membuat fitur multiuser/login dan register dipecah menjadi beberapa *task* sebagai berikut.

Tabel 4.25: *Sprint 11*

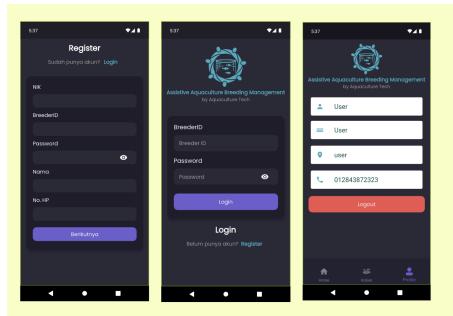
No	Story	Task	Status
1	Membuat multiuser	Membuat <i>Mock-up UI</i> halaman login dan register	selesai
2		Menerapkan <i>Mock-up UI</i> halaman login dan register ke Flutter	selesai
3		Mengintegrasikan halaman login dan register ke <i>webservice</i>	selesai

Pada sprint kesebelas ini story yang di pilih untuk di uraikan pada sprint kali ini adalah membuat login dan register. Tujuan dari *sprint-11* ini adalah membuat

multiuser dan mengintegrasikan halaman tersebut dengan webservice yang sudah dibuat oleh penelitian Andri Rahmanto.

1. Membuat Mock-up UI Fitur multiuser

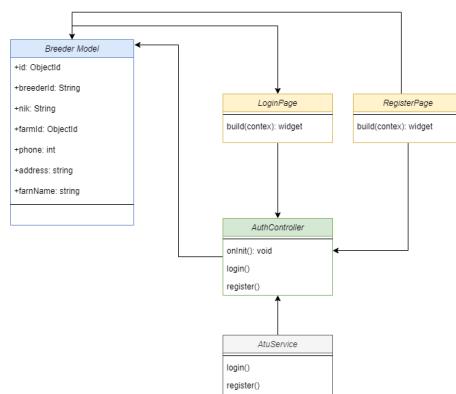
Pembuatan konten dan fitur yang terdapat pada *mock-up UI* fitur multiuser dilakukan berdasarkan persetujuan product owner dan scrum master pada meeting sebelumnya. Mock-up UI dibuat menggunakan platform figma.



Gambar 4.32: *Mock-up UI Fitur multiuser*

2. Class Diagram

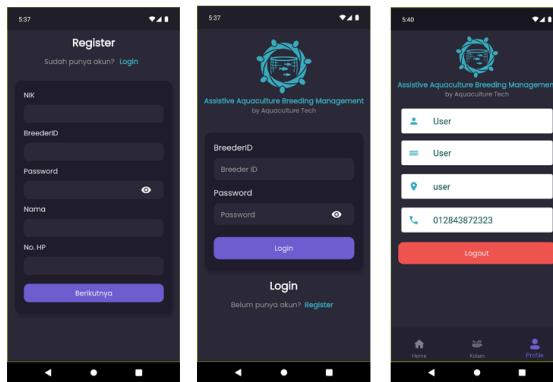
Class Diagram menggambarkan kelas-kelas yang akan dipakai oleh sistem. Umumnya terdapat 3 kelas pada setiap module yaitu class model, controller, dan view. Pada sprint-11 penelitian kali ini penulis membuat 4 class yaitu model yang berwarna biru, view berwarna oranye, controller yang berwarna hijau, dan service yang berwarna kuning.



Gambar 4.33: *Class Diagram Fitur Sprint-11*

3. Menerapkan Mockup-UI Fitur multiuser kedalam code flutter

Setelah itu, akan dilakukan pengimplementasian *mock-up UI* ke dalam aplikasi menggunakan flutter. Pada lampiran 12 terdapat source code dari implementasi fitur multiuser yang dikelompokan berdasarkan halaman yang menghasilkan output halaman seperti pada gambar dibawah ini.



Gambar 4.34: Output dari code pada sprint 11

4. Mengintegrasikan fitur multiuser dengan webservice

Hal yang dilakukan dalam mengintegrasikan fitur multiuser dengan webservice terpadat pada lampiran 12.

5. Analisis User Experience

Pada halaman register, pembudidaya harus memasukan data yang diperlukan untuk melakukan register sesuai dengan kesepakatan saat meeting. Setelah melakukan pendaftaran/register user dapat melakukan login dengan akun yang telah didaftarkan dengan menggunakan breeder ID dan password. Terdapat pula halaman profile yang berisi informasi yang terkait akun user/pembudidaya, dihalaman tersebut user dapat melakukan logout.

6. Sprint 11 Review

Sprint 11 diakhiri dengan melakukan weekly meeting pada hari selasa dengan agenda melakukan review dan testing terkait hasil sprint 11 dan melakukan planning untuk testing keseluruhan dengan rincian:

(a) *Review dan Testing hasil dari sprint 11*

Telah dilakukan review dan testing oleh penulis selaku developer dengan Scrum Master. Setelah dilakukan testing, Scrum Master menyimpulkan bahwa fitur multiuser telah berjalan dengan baik.

Tabel 4.26: Unit testing Halaman Awal.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Saat aplikasi dibuka akan muncul splash screen yang menampilkan logo yang menandakan aplikasi sedang loading	✓		Diterima
Setelah loading selesai maka akan ditampilkan halaman login	✓		Diterima

Tabel 4.27: Unit testing Halaman Login.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Ketika mengisi form login dengan data yang sesuai kemudian menekan submit, maka akan masuk ke halaman dashboard	✓		Diterima
Ketika mengisi form login dengan data yang tidak sesuai kemudian menekan submit, maka akan menampilkan pesan kesalahan	✓		Diterima

B. Pengujian Sistem

Pengujian sistem dilakukan menggunakan dua tipe yaitu Unit Testing dan User Acceptance Test (UAT). Pengujian dilaksanakan pada saat seluruh User Story pada Product Backlog telah diimplementasikan. Unit testing aplikasi dilakukan terhadap satu frontend developer, sedangkan pengujian User Acceptance Test dilakukan terhadap satu scrum master dan owner.

1. Unit Testing

Adapun hasil dari unit testing yang telah dilaksanakan kepada salah satu developer internal, pengujian tersebut dilakukan pada saat suatu fitur telah dinyatakan selesai yang dimana hasil unit testing tersebut telah dijelaskan pada masing masing sprint report sebelumnya. Kesimpulan dari unit testing yang telah dilakukan adalah fitur yang telah dibuat dapat berjalan dengan baik.

2. User Acceptance Testing

User Acceptance Test terhadap user dilaksanakan pada tanggal 22 April 2023 secara luring bertempat di kecamatan jasinga. Adapun hasil dari UAT yang telah dilaksanakan dapat dilihat pada tabel dibawah ini.

Tabel 4.28: User Acceptance Test

No	Acceptance Requirements	Kesesuaian			
		SS	S	TS	STS
1	Fitur login sudah sesuai dengan kebutuhan pembudidaya	✓			
2	Fitur dasboard sudah sesuai dengan kebutuhan pembudidaya	✓			
3	Fitur registrasi, aktivasi dan deaktivasi sudah sesuai dengan kebutuhan pembudidaya			✓	
4	Fitur pemberian dan rekapitulasi pakan sudah sesuai dengan kebutuhan pembudidaya		✓		
5	Fitur grading berat ikan sudah sesuai dengan kebutuhan pembudidaya	✓			
6	Fitur pencatatan kematian sudah sesuai dengan kebutuhan pembudidaya	✓			
7	Fitur treatment kolam sudah sesuai dengan kebutuhan pembudidaya	✓			
8	Fitur pencatatan kualitas air harian sudah sesuai dengan kebutuhan pembudidaya			✓	
9	Fitur pencatatan kualitas air mingguan sudah sesuai dengan kebutuhan pembudidaya			✓	
10	Fitur sortir ikan sudah sesuai dengan kebutuhan pembudidaya			✓	

Berdasarkan tabel diatas, terdapat empat fitur yang memerlukan revisi

diantaranya:

1. Pada fitur aktivasi kolam perlu ditambahkan input tipe aktivasi, yaitu pada saat melakukan aktivasi dapat memilih tipe aktivasinya dengan pilihan pembenihan dan pembesaran.
2. Pada fitur pencatatan kualitas air harian inputnya perlu diubah ke dalam bentuk desimal dikarenakan ukuran pH, Dissolved Oxygen. dan suhu dapat berupa desimal.
3. Pada fitur pencatatan kualitas air mingguan inputnya perlu diubah ke dalam format desimal dikarenakan data yang di input dapat berupa desimal.
4. Pada fitur sortir, terdapat perubahan flow dari fitur tersebut agar lebih sesuai dengan sortir ikan yang dilakukan oleh pembudidaya di lapangan.

3. Kesimpulan Pengujian

Pengujian aplikasi dilaksanakan dengan dua metode yaitu unit testing dan User Acceptance Test (UAT). Berdasarkan uraian di atas, seluruh skenario pada unit testing terhadap satu internal developer berjalan dengan baik. Namun, berdasarkan hasil UAT terhadap pembudidaya terdapat beberapa masukan seperti fitur aktivasi kolam, pencatatan kualitas air, dan fitur sortir kolam.

BAB V

KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan hasil implementasi dan pengujian fitur yang telah dirancang, maka diperoleh kesimpulan sebagai berikut:

1. Terciptanya aplikasi pendukung teknologi perikanan modern versi pertama yang sudah mengintegrasikan fitur-fitur pada Product Backlog. Adapun perancangannya dilakukan dengan metode Scrum dengan tahapan penyusunan Product Backlog, Sprint Backlog, dan dikerjakan dalam sebelas Sprint.
2. Berdasarkan hasil pengujian, seluruh skenario pada unit testing terhadap satu internal developer berjalan dengan baik. Namun, berdasarkan hasil UAT terhadap pembudidaya terdapat beberapa masukan seperti Pada fitur aktivasi kolam perlu ditambahkan input tipe aktivasi, apakah aktivasi tersebut atau pembesaran , pada fitur kualitas air inputnya perlu bisa dalam format desimal, dan pada fitur sortir, terdapat perubahan flow agar lebih sesuai dengan sortir ikan yang dilakukan oleh pembudidaya.

B. Saran

Adapun saran untuk penelitian selanjutnya adalah:

1. Berdasarkan diskusi dengan user/product owner, harus dimulainya pengembangan sistem berikutnya yang memiliki fitur inventarisasi dan penentuan harga jual ikan agar dapat membantu pembudidaya dalam menentukan harga hasil budayanya.
2. Berdasarkan diskusi dengan scrum master, perlu ditambahkan feedback dari sistem seperti pop up atau sejenisnya setelah berinteraksi dengan sistem, apakah interaksi yang dilakukan dengan sistem berhasil atau tidak.

DAFTAR PUSTAKA

- Avnimelech, Y. (2009). *Biofloc Technology-A Practical Guide Book*. The World Aquaculture Society, Louisiana.
- Bangkit, S. (2016). Pengembangan budidaya ikan air tawar berkreatif di karanganyar.
- Dalsgaard, J., Lund, I., Thorarinsdottir, R., Drengstig, A., Arvonen, K., dan Pedersen, P. (2015). Farming different species in ras in nordic countries: Current status and future perspectives. *Journal of Aquacultural Engineering*, 53:2–13.
- Effendie, I. (1997). *Biologi perikanan*. Yayasan Pustaka Nusatama, Yogyakarta.
- Fadhil, R., Johari, E., Farah, S., dan Salih, M. (2010). Teknologi sistem akuakultur resirkulasi untuk menenangkan produksi perikanan darat di aceh. *Aceh Development International Conference 2010*.
- Fattah, A. dan El-Sayed, M. (2020). *Tilapia Culture. 2nd Edition*. Elsevier, London.
- Keat, L. dan Chuah, C. W. (2018). Smart indoor home surveillance monitoring system using raspberry pi. *JOIV : International Journal on Informatics Visualization*, 2.
- Kelvin, A. dan Dian, G. (2021). Rancang bangun aplikasi berbasis android untuk brand clothing sand beach dengan skema diskon menggunakan hungarian algorithm. *Jurnal Sistem Informasi Universitas Surya Darmal*, 8(01).
- KKP (2022). Satu data kkp: Statistik produksi perikanan.
- Napoli (2019). Introducing flutter and getting started.
- Samocha, T. (2019). *Sustainable Biofloc System for Marine Shrim*. Elsevier, London.
- Setiawan, H. (2021). Teknologi bioflok dalam budidaya ikan nila. In *Training Bioflok Farm Tilapia*.
- Supriati dan Rizki, D. (2018). Model perancangan sistem informasi akuntansi budidaya perikanan berbasis sak emkm dan android. *Open Journal – Universitas Komputer Indonesia*, 3(02):301–3016.
- Sutherland, J. dan Schwaber, K. (2020). scrumguides.

- Sutherland, J., Viktorov, A., Blount, J., dan Puntikov, N. (2007). Distributed scrum: Agile project management with outsourced development teams. In *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*, pages 274a–274a.
- Taw, N. (2012). Recent development in biofloc technology biosecure system improve economics, sustaiability. *Global Aquac*, (01):28–29.
- Thesiana, L. dan Pamungkasi, A. (2015). Uji performansi teknologi recirculating aquaculture system (ras) terhadap kondisi kualitas air pada pendederan lobster pasir panulirus homarus. *Jurnal Kelautan Nasional*, 10(02):65–73.
- Widhiastika, D., Siroi, S., Umayati, A., Rafino, N., dan Putra, B. (2021). Perancangan aplikasi jual beli produk perikanan berbasis mobile android (studi kasus : Fo-klik). *JURNAL LEMURU*, 3(01):33–44.

LAMPIRAN

A. Lampiran 1 Transkrip Percakapan

Hari: Selasa

Tanggal: 23 Agustus 2022

P: Penulis

K: Klien (UD Jfarm)

P: Sistem apa yang akan dibuat?

K: Kita akan membuat frontend untuk aplikasi pendukung teknologi perikanan modern

P: Apa saja requirement yang diperlukan dalam aplikasi tersebut?

K: Requirementnya adalah penerapan fitur yang sudah ada dalam backend penelitian dari Andri Rahmanto yang berbentuk REST API.

P: Apa saja fitur yang ada dalam backend yang telah dibuat dalam penelitian Andri Rahmanto?

K: Fiturnya adalah pencatatan pemberian pakan, registrasi kolam, aktifasi-deaktifasi kolam, pencatatan kualitas air harian dan mingguan, pencatatan data kematian harian, pencatatan treatment kolam, grading berat ikan, dan sortir ikan.

B. Lampiran 2 code untuk sprint 1 report

```

Widget title() {
    return Container(
        margin: EdgeInsets.only(
            top: defaultMargin,
            left: defaultMargin,
            right: defaultMargin,
        ),
        child: Text(
            'Home',
            style: primaryTextStyle.copyWith(
                fontSize: 24,
                fontWeight: semiBold,
            ),
        ),
    );
}

Widget statistic() {
    return Container(
        margin: EdgeInsets.only(
            top: defaultMargin,
            left: defaultMargin,
        ),
        child: Column(
            crossAxisAlignment: CrossAxisAlignment.center,
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
                Row(
                    mainAxisAlignment: MainAxisAlignment.spaceBetween,
                    crossAxisAlignment: CrossAxisAlignment.center,
                    children: [
                        Expanded(
                            flex: 1,
                            child: StatisticCard(
                                title: 'Kolam',
                                value: controller.statistic.value.total_pond,
                            )),
                        Expanded(
                            flex: 1,
                            child: StatisticCard(
                                title: 'Kolam Aktif',
                                value: controller.statistic.value.active_pond,
                            )),
                    ],
                ),
                SizedBox(
                    height: 16,
                ),
                Row(
                    mainAxisAlignment: MainAxisAlignment.spaceBetween,

```

```

crossAxisAlignment: CrossAxisAlignment.Alignment.center,
children: [
  Expanded(
    flex: 1,
    child: StatisticCard(
      title: 'Ikan Hidup',
      value: controller.statistic.value.fish_live,
      unit: 'Ekor',
    )),
  Expanded(
    flex: 1,
    child: StatisticCard(
      title: 'Ikan Mati',
      value: controller.statistic.value.fish_death,
      unit: 'Ekor',
    )),
],
),
SizedBox(
  height: 16,
),
),
Row(
  mainAxisAlignment: MainAxisAlignment.spaceBetween,
  crossAxisAlignment: CrossAxisAlignment.Alignment.center,
  children: [
  Expanded(
    flex: 1,
    child: StatisticCard(
      title: 'Panen 2022',
      value: controller.statistic.value.fish_harvested,
      unit: 'Kg',
    )),
  Expanded(
    flex: 1,
    child: StatisticCard(
      title: 'Total Pakan',
      value: controller.statistic.value.total_feed_dose,
      unit: 'Kg',
    )),
],
),
],
);
}

```

Lampiran 3 code widget untuk section statistik ikan

```

Widget fishTitle() {
  return Container(
    margin: EdgeInsets.only(
      top: defaultSpace,
      left: defaultMargin,
    )
  );
}

```

```
        right: defaultMargin,
    ),
    child: Column(
        mainAxisAlignment: MainAxisAlignment.start,
        children: [
            Text(
                'Total Berat Ikan',
                style: primaryTextStyle.copyWith(
                    fontSize: 24,
                    fontWeight: semiBold,
                ),
            ),
        ],
    ),
),
);
}
}

Widget fish() {
return Container(
margin: EdgeInsets.only(top: 14),
child: SingleChildScrollView(
scrollDirection: Axis.horizontal,
child: Row(
children: [
SizedBox(
width: defaultMargin,
),
Row(children: [
FishCard(
title: "Lele",
value: controller.statistic.value.fishes_weight_lele!,
image: "assets/lele.png",
),
FishCard(
title: "Nila Merah",
value: controller.statistic.value.fishes_weight_nilamerah!,
image: "assets/nilamerah.png",
),
FishCard(
title: "Nila Hitam",
value: controller.statistic.value.fishes_weight_nilahitam!,
image: "assets/nilahitam.png",
),
FishCard(
title: "Mas",
value: controller.statistic.value.fishes_weight_mas!,
image: "assets/mas.png",
),
]),
],
),
),
);
}
```

}

code widget untuk section statistik kondisi air

```

Widget waterTitle() {
    return Container(
        margin: EdgeInsets.only(
            top: defaultSpace,
            left: defaultMargin,
            right: defaultMargin,
        ),
        child: Column(
            mainAxisAlignment: MainAxisAlignment.start,
            children: [
                Text(
                    'Kualitas Air',
                    style: primaryTextStyle.copyWith(
                        fontSize: 24,
                        fontWeight: semiBold,
                    ),
                ),
                ],
            ),
        );
    }
}

Widget water() {
    return Container(
        margin: EdgeInsets.only(top: 14),
        child: SingleChildScrollView(
            scrollDirection: Axis.horizontal,
            child: Row(
                children: [
                    SizedBox(
                        width: defaultMargin,
                    ),
                    Row(children: [
                        WaterCard(
                            title: "pH",
                            normal: controller.statistic.value.ph_normal,
                            abnormal: controller.statistic.value.ph_abnormal,
                        ),
                        WaterCard(
                            title: "DO",
                            normal: controller.statistic.value.do_normal,
                            abnormal: controller.statistic.value.do_abnormal,
                        ),
                        WaterCard(
                            title: "Flok",
                            normal: controller.statistic.value.floc_normal,
                            abnormal: controller.statistic.value.floc_abnormal,
                        ),
                    ],
                ],
            ),
        );
    }
}

```

```

        ],
    ],
),
),
);
}

```

code widget untuk bagian bottom navigation bar

```

bottomNavigationBar: BottomNavigationBar(
    type: BottomNavigationBarType.fixed,
    backgroundColor: backgroundColor3,
    onTap: controller.changeTabIndex,
    currentIndex: controller.tabIndex,
    items: [
        BottomNavigationBarItem(
            icon: Container(
                margin: EdgeInsets.only(
                    top: 20,
                    bottom: 5,
                ),
                child: Image.asset(
                    'assets/home_secondary.png',
                    width: 25,
                    color: controller.tabIndex == 0
                        ? primaryColor
                        : Color(0xff808191),
                ),
            ),
            label: '',
        ),
        BottomNavigationBarItem(
            icon: Container(
                margin: EdgeInsets.only(
                    top: 20,
                    bottom: 5,
                ),
                child: Image.asset(
                    'assets/pond_secondary.png',
                    width: 25,
                    color: controller.tabIndex == 1
                        ? primaryColor
                        : Color(0xff808191),
                ),
            ),
            label: '',
        ),
    ],
),

```

Mengintegrasikan dengan webservice

Model Class untuk Halaman Dashboard

```

class StatisticModel {
    int? total_pond;
    int? active_pond;
    int? fish_live;
    int? fish_death;
    int? fish_harvested;
    int? total_feed_dose;
    num? fishes_weight_lele;
    num? fishes_weight_nilamerah;
    num? fishes_weight_nilahitam;
    num? fishes_weight_mas;
    num? fishes_weight_patin;
    int? ph_normal;
    int? ph_abnormal;
    int? do_normal;
    int? do_abnormal;
    int? floc_normal;
    int? floc_abnormal;

    StatisticModel(
        this.total_pond,
        this.active_pond,
        this.fish_live,
        this.fish_death,
        this.fish_harvested,
        this.total_feed_dose,
        this.fishes_weight_lele,
        this.fishes_weight_nilamerah,
        this.fishes_weight_nilahitam,
        this.fishes_weight_mas,
        this.fishes_weight_patin,
        this.ph_normal,
        this.ph_abnormal,
        this.do_normal,
        this.do_abnormal,
        this.floc_normal,
        this.floc_abnormal));

    StatisticModel.fromJson(Map<String, dynamic> json) {
        total_pond = json['total_pond'];
        active_pond = json['active_pond'];
        fish_live = json['fish_live'];
        fish_death = json['fish_death'];
        fish_harvested = json['fish_harvested'];
        total_feed_dose = json['total_feed_dose'];
        fishes_weight_nilahitam = json['fishes_weight'][0]["amount"];
        fishes_weight_nilamerah = json['fishes_weight'][1]["amount"];
        fishes_weight_lele = json['fishes_weight'][2]["amount"];
        fishes_weight_patin = json['fishes_weight'][3]["amount"];
        fishes_weight_mas = json['fishes_weight'][4]["amount"];
        ph_normal = json['water_quality']['ph']['normal'];
    }
}

```

```

    ph_abnormal = json['water_quality']['ph']['abnormal'];
    do_normal = json['water_quality']['do']['normal'];
    do_abnormal = json['water_quality']['do']['abnormal'];
    floc_normal = json['water_quality']['floc']['normal'];
    floc_abnormal = json['water_quality']['floc']['abnormal'];
}
}

```

Networt Request untuk Halaman Dashboard

```

import 'dart:convert';
import 'package:fish/models/statistic_model.dart';
import 'package:fish/service/url_api.dart';
import 'package:http/http.dart' as http;

class StatisticService {
  Future<StatisticModel> getStatistic() async {
    var url = Uri.parse('http://jft.web.id/fishapi/api/statistic');
    var headers = {'Content-Type': 'application/json'};

    var response = await http.get(url, headers: headers);

    print(response.body);

    if (response.statusCode == 200) {
      var data = jsonDecode(response.body);
      StatisticModel statistic = StatisticModel.fromJson(data);

      return statistic;
    } else {
      throw Exception('Gagal Get Products!');
    }
  }
}

```

Mengolah data yang telah direquest pada Controller

```

class HomeController extends GetxController {
  var isLoading = false.obs;

  final statistic = StatisticModel().obs;

  @override
  void onInit() async {
    await getStatisticData();
    super.onInit();
  }

  Future<void> getStatisticData() async {
    isLoading.value = true;

```

```

StatisticModel statisticData = await StatisticService().getStatistic();
statistic.value = statisticData;
Timer(const Duration(seconds: 1), () {
  isLoading.value = false;
});
}
}

```

Menampilkan data pada Halaman Dashboard

```

Widget water() {
  return Container(
    margin: EdgeInsets.only(top: 14),
    child: SingleChildScrollView(
      scrollDirection: Axis.horizontal,
      child: Row(
        children: [
          SizedBox(
            width: defaultMargin,
          ),
          Row(children: [
            WaterCard(
              title: "pH",
              normal: controller.statistic.value.ph_normal,
              abnormal: controller.statistic.value.ph_abnormal,
            ),
            WaterCard(
              title: "DO",
              normal: controller.statistic.value.do_normal,
              abnormal: controller.statistic.value.do_abnormal,
            ),
            WaterCard(
              title: "Flok",
              normal: controller.statistic.value.floc_normal,
              abnormal: controller.statistic.value.floc_abnormal,
            ),
          ]),
        ],
      ),
    );
}

```

C. Lampiran 3 Sprint 2 report

```

import 'package:fish/models/pond_model.dart';
import 'package:fish/pages/component/pond_card.dart';

import 'package:fish/pages/pond/add_pond_page.dart';
import 'package:fish/pages/pond/pond_controller.dart';
import 'package:flutter/material.dart';
import 'package:fish/theme.dart';
import 'package:get/get.dart';

class PondPage extends StatefulWidget {
    PondPage({Key? key}) : super(key: key);
    @override
    State<PondPage> createState() => _PondPageState();
}

class _PondPageState extends State<PondPage> {
    final PondController controller = Get.put(PondController());
    int? _value = null;
    final chip = ["Aktif", "Panen", "Tidak Aktif"];
    @override
    void initState() {
        super.initState();

        controller.getPondsData(context);
    }

    @override
    Widget build(BuildContext context) {
        Widget title() {
            return Container(
                margin: EdgeInsets.only(
                    top: defaultMargin,
                    left: defaultMargin,
                    right: defaultMargin,
                ),
                child: Text(
                    'Kolam',
                    style: primaryTextStyle.copyWith(
                        fontSize: 24,
                        fontWeight: semiBold,
                    ),
                ),
            );
        }
        Widget filter() {
            return Container(
                margin: EdgeInsets.only(
                    top: defaultMargin,
                    left: defaultMargin,
                ),
            );
        }
    }
}

```

```

        right: defaultMargin,
),
child: Wrap(
    spacing: 8.0,
    children: List<Widget>.generate(
        3,
        (int index) {
            return ChoiceChip(
                label: Text(
                    chip[index],
                    style: TextStyle(color: Colors.white),
                ),
                shape: StadiumBorder(side: BorderSide(color: Colors.white)),
                selected: _value == index,
                backgroundColor: backgroundColor1,
                selectedColor: primaryColor,
                onSelected: (bool selected) {
                    setState(() {
                        _value = selected ? index : null;
                    if (_value == null) {
                        controller.getPondsData(context);
                        // return null;
                    } else {
                        controller.getPondsFiltered(chip[index]);
                    }
                });
            },
        );
    },
).toList(),
),
);
}
}

Widget pondList() {
return Container(
margin: EdgeInsets.only(top: 14),
child: SingleChildScrollView(
child: ListView.builder(
shrinkWrap: true,
primary: false,
itemBuilder: ((context, index) {
return PondCard(pond: controller.ponds[index]
// indicatorWater: controller.indicatorWater[index]);
);
}),
itemCount: controller.ponds.length,
),
),
);
}
}

Widget emptyListPond() {

```

```

    return Container(
        width: double.infinity,
        margin: EdgeInsets.only(right: defaultMargin, left: defaultMargin),
        child: Center(
            child: Column(children: [
                SizedBox(height: 35),
                Image(
                    image: AssetImage("assets/unavailable_icon.png"),
                    width: 100,
                    height: 100,
                    fit: BoxFit.fitWidth,
                ),
                SizedBox(height: 20),
                Text(
                    "Anda belum pernah melakukan registrasi kolam",
                    style: primaryTextStyle.copyWith(
                        fontSize: 14,
                        fontWeight: bold,
                    ),
                    textAlign: TextAlign.center,
                    overflow: TextOverflow.ellipsis,
                    maxLines: 2,
                ),
                SizedBox(height: 10),
                Text(
                    "Silahkan registrasi kolam",
                    style: secondaryTextStyle.copyWith(
                        fontSize: 13,
                        fontWeight: bold,
                    ),
                    textAlign: TextAlign.center,
                    overflow: TextOverflow.ellipsis,
                    maxLines: 2,
                ),
            ]),
        )));
    });

    return Obx(() {
        if (controller.isLoading.value == false) {
            return Scaffold(
                backgroundColor: backgroundColor1,
                floatingActionButton: FloatingActionButton(
                    onPressed: () {
                        Get.to(() => AddPondPage());
                    },
                    backgroundColor: primaryColor,
                    child: const Icon(Icons.add),
                ),
                body: ListView(
                    children: [
                        title(),
                        filter(),

```

```

        controller.ponds.isEmpty ? emptyListPond() : pondList(),
        SizedBox(
          height: 10,
        )
      ],
    ),
  );
} else {
  return Center(
    child: CircularProgressIndicator(
      color: secondaryColor,
    ),
  );
}
));
}
}
}

```

Code widget untuk registrasi kolam

```

import 'package:fish/pages/pond/pond_controller.dart';
import 'package:flutter/material.dart';
import 'package:fish/theme.dart';
import 'package:flutter/services.dart';
import 'package:get/get.dart';

class AddPondPage extends StatefulWidget {
  const AddPondPage({Key? key}) : super(key: key);

  @override
  State<AddPondPage> createState() => _AddPondPageState();
}

class _AddPondPageState extends State<AddPondPage> {
  final PondController controller = Get.put(PondController());

  @override
  void dispose() {
    controller.aliasController.clear();
    controller.diameterController.clear();
    controller.heightController.clear();
    controller.locationController.clear();
    controller.lengthController.clear();
    controller.widthController.clear();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    Widget aliasInput() {...}
    Widget locationInput() {...}
    Widget materialInput() {...}
  }
}

```

```

Widget shapeInput() {...}
Widget heightInput() {...}
Widget lengthInput() {...}
Widget widthInput() {...}
Widget diameterInput() {...}
Widget registerButton() {...}
Widget persegiInput() {...}
Widget bundarInput() {...}

return Obx(() {
  if (controller.isLoading.value == false) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: backgroundColor2,
        title: const Text("Registrasi Kolam"),
      ),
      backgroundColor: backgroundColor1,
      body: ListView(
        children: [
          aliasInput(),
          locationInput(),
          materialInput(),
          shapeInput(),
          controller.shapeController.selected.value == 'persegi'
            ? persegiInput()
            : bundarInput(),
          heightInput(),
          registerButton(),
          SizedBox(
            height: 8,
          )
        ],
      ),
    );
  } else {
    return Center(
      child: CircularProgressIndicator(
        color: secondaryColor,
      ),
    );
  }
));
}
}

```

Code widget untuk halaman detail kolam

```

import 'dart:developer';

import 'package:fish/models/pond_model.dart';
import 'package:fish/pages/component/activation_card.dart';
import 'package:fish/pages/pond/activation_breed_controller.dart';

```

```
import 'package:fish/pages/pond/activation_breed_page.dart';
import 'package:fish/pages/pond/pond_controller.dart';
import 'package:fish/pages/pond/add_pond_page.dart';
import 'package:fish/pages/pond/deactivation_breed_page.dart';
import 'package:fish/pages/pond/detail_pond_controller.dart';
import 'package:flutter/material.dart';
import 'package:fish/theme.dart';
import 'package:get/get.dart';

import '../fish_transfer/fish_transfer_entry_page.dart';

class DetailPondPage extends StatefulWidget {
const DetailPondPage({Key? key}) : super(key: key);

@Override
State<DetailPondPage> createState() => _DetailPondPageState();
}

class _DetailPondPageState extends State<DetailPondPage> {
var detailController = Get.put(DetailPondController(), permanent: false);
@Override
void initState() {
super.initState();

detailController.getPondActivation(context);
}

@Override
void activate() {
print('ini aktif');
super.activate();
}

@Override
void deactivate() {
print('ini deaktif');
super.deactivate();
}

@Override
Widget build(BuildContext context) {
Widget pondStatus() {...}

Widget activationButton() {...}

Widget deactivationButton() {...}

Widget detail() {...}

Widget activationTitle() {...}

Widget listActivation() {...}
```

```

Widget emptyListActivation() {...}

return Scaffold(
  backgroundColor: backgroundColor1,
  body: Obx(
    () => detailController.isLoading.value
      ? Center(
          child: CircularProgressIndicator(
            color: secondaryColor,
          ),
        )
      : ListView(
          children: [
            pondStatus(),
            detailController.isPondActive.value == false
              ? activationButton()
              : deactivationButton(),
            detail(),
            activationTitle(),
            detailController.activations.isEmpty
              ? emptyListActivation()
              : listActivation(),
            SizedBox(
              height: 10,
            )
          ],
        ),
  ),
);
}
}

```

Code halaman aktivasi kolam

```

import 'dart:developer';

import 'package:fish/pages/pond/activation_breed_controller.dart';
import 'package:fish/pages/pond/detail_pond_controller.dart';
import 'package:fish/pages/pond/detail_pond_page.dart';
import 'package:fish/service/pond_service.dart';
import 'package:fish/service/activation_service.dart';
import 'package:flutter/material.dart';
import 'package:fish/theme.dart';
import 'package:flutter/services.dart';
import 'package:get/get.dart';

import '../component/detail_pond_tabview.dart';

class ActivationBreedPage extends StatelessWidget {
  ActivationBreedPage({Key? key}) : super(key: key);

  final ActivationBreedController controller =

```

```
Get.put(ActivationBreedController());  
  
final DetailPondController detailPondController =  
    Get.put(DetailPondController());  
  
@override  
Widget build(BuildContext context) {  
    Widget checkBoxFish() {...}  
  
    Widget waterHeightInput() {...}  
  
    Widget leleInput() {...}  
  
    Widget nilaMerahInput() {...}  
  
    Widget nilaHitamInput() {...}  
  
    Widget patinInput() {...}  
  
    Widget masInput() {...}  
  
    Widget breedOptionInput() {...}  
  
    Widget pembesaranInput() {...}  
  
    Widget benihInput() {...}  
  
    Widget activationButton() {...}  
  
    return Obx(() {  
        if (controller.isActivationProgress.value == false) {  
            return Scaffold(  
                appBar: AppBar(  
                    backgroundColor: backgroundColor2,  
                    title: const Text("Aktivasi Kolam"),  
                ),  
                backgroundColor: backgroundColor1,  
                body: ListView(  
                    children: [  
                        breedOptionInput(),  
                        controller.breedOptionController.selected.value == "Benih"  
                            ? benihInput()  
                            : pembesaranInput(),  
                        checkBoxFish(),  
                        controller.isNilaHitam == true ? nilaHitamInput() : Container(),  
                        controller.isNilaMerah == true ? nilaMerahInput() : Container(),  
                        controller.isLele == true ? leleInput() : Container(),  
                        controller.isPatin == true ? patinInput() : Container(),  
                        controller.isMas == true ? masInput() : Container(),  
                        waterHeightInput(),  
                        activationButton(),  
                        SizedBox(  
                            height: 8,
```

```
)  
],  
)  
} else {  
    return Center(  
        child: CircularProgressIndicator(  
            color: secondaryColor,  
        ),  
    );  
}  
});  
}  
}
```

D. Lampiran 4 Code Sprint 3 report

Code untuk model class kolam

```

import 'package:flutter/material.dart';
import 'package:intl/intl.dart';

enum PondStatus {
  active,
  nonActive,
  close,
}

class string {}

class Pond {
  String? id;
  int? idInt;
  String? alias;
  String? location;
  String? shape;
  String? material;
  num? length;
  num? width;
  num? diameter;
  num? height;
  num? area;
  num? volume;
  String? buildAt;
  String? imageLink;
  bool? isActive;
  num? fishAlive;
  String? lastActivationDate;
  String? rangeFromLastActivation;
  PondStatus pondStatus;
  String? pondStatusStr;
  String? pondPhDesc;
  num? pondPh;
  String? pondDoDesc;
  num? pondDo;
  num? pondTemp;
  String? status;

  Pond({
    required this.id,
    required this.idInt,
    required this.alias,
    required this.location,
    required this.shape,
    required this.material,
    required this.isActive,
    required this.pondStatus,
  })
}

```

```

    this.length,
    this.width,
    this.diameter,
    this.height,
    this.area,
    this.volume,
    this.buildAt,
    this.imageLink,
    this.fishAlive,
    this.lastActivationDate,
    this.rangeFromLastActivation,
    this.pondStatusStr,
    this.pondPh,
    this.pondPhDesc,
    this.pondDo,
    this.pondDoDesc,
    this.pondTemp,
    this.status,
  });

factory Pond.fromJson(Map<String, dynamic> json) {
  return Pond(
    id: json['_id'],
    idInt: json['id_int'],
    alias: json['alias'],
    location: json['location'],
    shape: json['shape'],
    material: json['material'],
    length: json['length'],
    width: json['width'],
    diameter: json['diameter'],
    height: json['height'],
    area: json['area'],
    volume: json['volume'],
    buildAt: json['build_at'],
    imageLink: json['image_link'],
    isActive: json['isActive'],
    fishAlive: json['fish_alive'] ?? 0,
    lastActivationDate: json['activation_date'] ?? "-",
    rangeFromLastActivation: json['isActive'] == false
      ? "-"
      : (DateTime.now()
          .difference(stringToDate(json['activation_date']))
          .inDays)
        .toString(),
    pondStatus: PondStatusConverter.toEnum(json["status"]),
    pondStatusStr: json["status"],
    pondPh: json["pondPh"],
    pondPhDesc: json["pondPhDesc"],
    pondDo: json["pondDo"],
    pondDoDesc: json["pondDoDesc"],
    pondTemp: json["pondTemp"],
    status: json["status"]);
}

```

```

}

static DateTime stringToDate(String dateString) {
    DateTime parseDate = DateFormat("dd-MM-yyyy").parse(dateString);
    return parseDate;
}

Color getColor() {
    PondStatus pondStatus = this.pondStatus;
    switch (pondStatus) {
        case PondStatus.active:
            return Colors.green;
        case PondStatus.nonActive:
            return Colors.red.shade300;
        case PondStatus.close:
            return Colors.amber;
        default:
            return Colors.red.shade300;
    }
}

String getFishAlive() {
    if (isActive == false) {
        return '-';
    } else {
        return fishAlive.toString();
    }
}

String getLastActivationDate() {
    if (isActive == false) {
        return '-';
    } else {
        return lastActivationDate!;
    }
}

String getGmtToNormalDate() {
    String stringDate = buildAt!;
    DateTime dateTime = DateFormat("yyyy-MM-dd hh:mm:ss").parse(stringDate);
    String newStringDate = DateFormat("dd-MM-yyyy").format(dateTime);
    return newStringDate;
}
}

extension PondStatusConverter on PondStatus {
    static PondStatus toEnum(String? status) {
        switch (status) {
            case 'Aktif':
                return PondStatus.active;
            case 'Tidak Aktif':
                return PondStatus.nonActive;
            case 'Panen':

```

```

        return PondStatus.close;
    default:
        return PondStatus.nonActive;
    }
}
}

```

Code untuk API service halaman terkait kolam kolam

```

import 'dart:convert';
import 'dart:math';
import 'package:fish/models/pond_model.dart';
import 'package:fish/service/url_api.dart';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'package:shared_preferences/shared_preferences.dart';

import '../theme.dart';

class PondService {
    Future<List<Pond>> getPonds() async {
        WidgetsFlutterBinding.ensureInitialized();
        SharedPreferences prefs = await SharedPreferences.getInstance();
        String token = prefs.getString('token').toString();
        var url = Uri.parse(Urls.ponds);
        var headers = {
            'Content-Type': 'application/json',
            'Authorization': 'Bearer $token'
        };

        var response = await http.get(url, headers: headers);

        print(response.body);

        if (response.statusCode == 200) {
            var data = jsonDecode(response.body);
            List<Pond> ponds = [];

            for (var item in data) {
                ponds.add(Pond.fromJson(item));
            }

            print(ponds);

            return ponds;
        } else {
            throw Exception(e);
        }
    }

    Future<void> getPondDetail({required String pondId}) async {
        var url = Uri.parse(Urls.pond(pondId));
    }
}

```

```

var headers = {'Content-Type': 'application/json'};

var response = await http.get(url, headers: headers);

print(response.body);

if (response.statusCode == 200) {
  var data = jsonDecode(response.body);
  // Pond pond = Pond.fromJson(data);
  // print(pond);

  // return pond;
} else {
  throw Exception('Gagal Get Detial Pond!');
}
}

Future<bool> pondRegister(
  required String? alias,
  required String? location,
  required String? shape,
  required String? material,
  required String? length,
  required String? width,
  required String? diameter,
  required String? height,
  required String? status,
  required Function doInPost,
  required BuildContext context)) async {
  if (diameter!.isNotEmpty) {
    if (diameter.contains(",")) {
      diameter = diameter.replaceAll(',', '.');
    }
  }
  if (length!.isNotEmpty) {
    if (length.contains(",")) {
      length = length.replaceAll(',', '.');
    }
  }
  if (width!.isNotEmpty) {
    if (width.contains(",")) {
      width = width.replaceAll(',', '.');
    }
  }
  if (height!.isNotEmpty) {
    if (height.contains(",")) {
      height = height.replaceAll(',', '.');
    }
  }
  WidgetsFlutterBinding.ensureInitialized();
  SharedPreferences prefs = await SharedPreferences.getInstance();
  String token = prefs.getString('token').toString();
  final response = await http.post(

```

```

Uri.parse(Urls.ponds),
headers: {
  "Content-Type": "application/x-www-form-urlencoded",
  'Authorization': 'Bearer $token'
},
encoding: Encoding.getByName('utf-8'),
body: {
  "alias": alias,
  "location": location,
  "shape": shape,
  "material": material,
  "status": status,
  "length": length,
  "width": width,
  "diameter": diameter,
  "height": height,
},
);

if (response.statusCode == 200) {
  doInPost();
  Navigator.pop(context);
  return true;
} else {
  var res = jsonDecode(response.body);

  showDialog<String>(
    context: context,
    builder: (BuildContext context) => AlertDialog(
      title: const Text('Input Error',
        style: TextStyle(color: Colors.red)),
      content: Text(
        '${res['message']}',
        style: TextStyle(color: Colors.white),
      ),
      backgroundColor: backgroundColor1,
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.all(Radius.circular(16.0))),
      actions: <Widget>[
        TextButton(
          onPressed: () => Navigator.pop(context, 'OK'),
          child: const Text('OK'),
        ),
      ],
    )));
  return false;
}
}

```

Code widget untuk halaman pemberian pakan

```
import 'package:fish/models/feed_chart_model.dart';
import 'package:fish/pages/component/feed_month_card.dart';
import 'package:fish/pages/feeding/feed_controller.dart';
import 'package:fish/pages/pond/detail_pond_controller.dart';
import 'package:fish/pages/pond/pond_controller.dart';
import 'package:flutter/material.dart';
import 'package:fish/pages/feeding/feed_entry_page.dart';
import 'package:fish/theme.dart';
import 'package:get/get.dart';
import 'package:syncfusion_flutter_charts/charts.dart';

class DetailFeedPage extends StatelessWidget {
  const DetailFeedPage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final FeedController controller = Get.put(FeedController());
    final PondController pondController = Get.find();
    final DetailPondController detailPondController = Get.find();

    Widget chartFeed() {...}

    Widget emptyListPond() {...}

    Widget feedDataRecap() {...}

    Widget entryPakanButton() {...}

    Widget detail() {...}

    Widget recapTitle() {...}

    Widget listMonthFeed() {...}

    return Obx(() {
      if (controller.isLoading.value == false) {
        return Scaffold(
          appBar: AppBar(
            backgroundColor: backgroundColor2,
            title: const Text("Detail Pakan Permusim"),
          ),
          backgroundColor: backgroundColor1,
          body: ListView(
            children: [
              chartFeed(),
              feedDataRecap(),
              // detail(),
              entryPakanButton(),
              recapTitle(),
              // chartRecap(),
              controller.list_feedHistoryMonthly.isEmpty
                ? emptyListPond()
                : listMonthFeed(),
            ],
          ),
        );
      } else {
        return Center(
          child: CircularProgressIndicator(),
        );
      }
    });
  }
}
```

```
        : listMonthFeed(),
      SizedBox(
        height: 10,
      )
    ],
  ),
),
);
} else {
  return Center(
    child: CircularProgressIndicator(
      color: secondaryColor,
    ),
  );
}
});  
}  
}
```

Code untuk halaman entry pakan

```

),
Container(
  height: 50,
  padding: EdgeInsets.symmetric(
    horizontal: 16,
  ),
decoration: BoxDecoration(
  color: backgroundColor2,
  borderRadius: BorderRadius.circular(12),
),
child: Center(
  child: Obx(() => DropdownButtonFormField<String>(
    onChanged: (newValue) => controller.feedTypeFormController
      .setSelected(newValue!),
    value: controller.feedTypeFormController.selected.value,
    items: controller.feedTypeFormController.listFeedType
      .map((feedtype) {
        return DropdownMenuItem<String>(
          value: feedtype.type,
          child: Text(
            feedtype.type.toString(),
            style: primaryTextStyle,
          ),
        );
      })
      .toList(),
    dropdownColor: backgroundColor5,
    decoration: InputDecoration(border: InputBorder.none),
  )),
),
),
],
),
);
);
}

Widget feedSatuanInput() {
return Container(
margin: EdgeInsets.only(
  top: defaultSpace, right: defaultMargin, left: defaultMargin),
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
Text(
  'Pilih Satuan',
  style: primaryTextStyle.copyWith(
    fontSize: 16,
    fontWeight: medium,
  ),
),
SizedBox(
  height: 12,
),
Container(

```

```

height: 50,
padding: EdgeInsets.symmetric(
    horizontal: 16,
),
decoration: BoxDecoration(
    color: backgroundColor2,
    borderRadius: BorderRadius.circular(12),
),
child: Center(
    child: Obx(() => DropdownButtonFormField<String>(
        onChanged: (newValue) => controller.feedSatuanController
            .setSelected(newValue!),
        value: controller.feedSatuanController.selected.value,
        items: controller.feedSatuanController.listSatuan
            .map((feedtype) {
                return DropdownMenuItem<String>(
                    value: feedtype,
                    child: Text(
                        feedtype.toString(),
                        style: primaryTextStyle,
                    ),
                );
            }).toList(),
        dropdownColor: backgroundColor5,
        decoration: InputDecoration(border: InputBorder.none),
    )),
),
),
],
),
);
}
}

Widget feedDosisInput() {
    return Container(
        margin: EdgeInsets.only(
            top: defaultSpace, right: defaultMargin, left: defaultMargin),
        child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
                Text(
                    'Dosis Pakan',
                    style: primaryTextStyle.copyWith(
                        fontSize: 16,
                        fontWeight: medium,
                    ),
                ),
                SizedBox(
                    height: 12,
                ),
                Container(
                    height: 50,
                    padding: EdgeInsets.symmetric(

```

```

        horizontal: 16,
    ),
    decoration: BoxDecoration(
        color: backgroundColor2,
        borderRadius: BorderRadius.circular(12),
    ),
    child: Center(child: Obx(() {
        return TextFormField(
            style: primaryTextStyle,
            inputFormatters: <TextInputFormatter>[
                FilteringTextInputFormatter.deny(RegExp(r'[-+=*#%/, \s]+'))
            ],
            keyboardType: TextInputType.number,
            onChanged: controller.doseChanged,
            onTap: controller.valdose,
            controller: controller.feedDosisController,
            decoration: controller.validatedose.value == true
                ? controller.dose == ''
                    ? InputDecoration(
                        errorText: 'Dosis tidak boleh kosong',
                        isCollapsed: true)
                    : null
                : InputDecoration.collapsed(
                    hintText: 'ex: 2.1', hintStyle: subtitleTextStyle),
        );
    })),
),
],
),
);
}
}

Widget submitButton() {
    return Container(
        height: 50,
        width: double.infinity,
        margin: EdgeInsets.only(
            top: defaultSpace * 3, right: defaultMargin, left: defaultMargin),
        child: TextButton(
            onPressed: () async {
                controller.feedDosisController.text == ""
                    ? null
                    : Navigator.pop(context);
                controller.postFeedHistory();
                feedcontroller.getChartFeed(
                    activation_id: controller.activation.id.toString());
                feedcontroller.getWeeklyRecapFeedHistory(
                    activation_id: controller.activation.id.toString());
                controller.postDataLog(controller.fitur);
            },
            style: TextButton.styleFrom(
                backgroundColor: primaryColor,
                shape: RoundedRectangleBorder(

```

```

        borderRadius: BorderRadius.circular(12),
    ),
),
child: Text(
    'Submit',
    style: primaryTextStyle.copyWith(
        fontSize: 16,
        fontWeight: medium,
    ),
),
),
),
);
}
}

return Obx(() {
if (controller.isLoading.value == false) {
return Scaffold(
appBar: AppBar(
backgroundColor: backgroundColor2,
title: Text("Entry Pakan Kolam ${controller.pond.alias}"),
),
backgroundColor: backgroundColor1,
body: ListView(
children: [
// pondInput(),
feedTypeInput(),
feedSatuanInput(),
feedDosisInput(),
submitButton(),
SizedBox(
height: 8,
)
],
),
),
);
} else {
return Center(
child: CircularProgressIndicator(
color: secondaryColor,
),
);
}
));
}
}

```

Membuat model class untuk fitur pemberian pakan

```

import 'package:intl/intl.dart';

class FeedHistoryMonthly {
DateTime? date;

```

```

num? totalFeedWeight;
num? totalFeed;

FeedHistoryMonthly({
  required this.date,
  required this.totalFeedWeight,
  required this.totalFeed,
});

factory FeedHistoryMonthly.fromJson(Map<String, dynamic> json) {
  return FeedHistoryMonthly(
    date: DateTime.utc(json['year'], json['_id']),
    totalFeedWeight: json['total_feed'],
    totalFeed: json['total_feedhistory'],
  );
}

static List<FeedHistoryMonthly> fromJsonList(List<dynamic> list) {
  List<FeedHistoryMonthly> fishes = [];
  for (var item in list) {
    fishes.add(FeedHistoryMonthly.fromJson(item));
  }
  return fishes;
}

String getMonthName() => DateFormat('MMM').format(date!);
String getMonthNameFull() => DateFormat('MMMM').format(date!);

String getMonth() => DateFormat('yyyy-MM').format(date!);
}

```

Membuat network request untuk fitur pemberian pakan

```

import 'dart:developer';
import 'dart:convert';
import 'package:fish/models/FeedHistoryDaily.dart';
import 'package:fish/models/FeedHistoryHourly.dart';
import 'package:fish/models/FeedHistoryMonthly.dart';
import 'package:fish/models/FeedHistoryWeekly.dart';
import 'package:fish/models/feed_chart_model.dart';
import 'package:fish/service/url_api.dart';
import 'package:http/http.dart' as http;

class FeedHistoryService {
  Future<List<FeedChartData>> getChart({required String activation_id}) async {
    var url = Uri.parseUrls.feedChartApi(activation_id));
    print(url);
    var headers = {'Content-Type': 'application/json'};

    var response = await http.get(url, headers: headers);

    print(response.body);
  }
}

```

```

if (response.statusCode == 200) {
    var data = jsonDecode(response.body);
    List<FeedChartData> feedChartData = FeedChartData.fromJsonList(data);
    return feedChartData;
} else {
    throw Exception('Gagal Get Activation!');
}
}

Future<bool> postFeedHistory({
    required String? pondId,
    required String? feedTypeId,
    required String? feedDose,
}) async {
    print({
        "pond_id": pondId,
        "feed_type_id": feedTypeId,
        "feed_dose": feedDose,
    });
    final response = await http.post(
        Uri.parse(Urls.feedhistory),
        headers: {
            "Content-Type": "application/x-www-form-urlencoded",
        },
        encoding: Encoding.getByName('utf-8'),
        body: {
            "pond_id": pondId,
            "feed_type_id": feedTypeId,
            "feed_dose": feedDose,
        },
    );
}

if (response.statusCode == 200) {
    print(response.body);
    return true;
} else {
    print(response.body);
    return false;
}
}
}

```

E. Lampiran 5 Code Sprint 4 report

Code halaman rekapitulasi grading

```

import 'dart:async';
import 'package:fish/models/grading_chart_model.dart';
import 'package:fish/pages/component/grading_card.dart';
import 'package:fish/pages/grading/grading_controller.dart';
import 'package:flutter/material.dart';
import 'package:fish/pages/grading/grading_constants_edit_page.dart';
import 'package:fish/pages/grading/grading_entry_page.dart';
import 'package:fish/theme.dart';
import 'package:get/get.dart';
import 'package:syncfusion_flutter_charts/charts.dart';

class GradingPage extends StatelessWidget {
  const GradingPage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final GradingController controller = Get.put(GradingController());

    Widget chartGrading() {...}

    Widget gradingDataRecap() {...}

    Widget entryGradingButton() {...}

    Widget detail() {...}

    Widget recapTitle() {...}

    Widget emptyListGrading() {...}

    Widget listGrading() {...}

    Widget sizingSec() {...}

    return Obx(() {
      if (controller.isLoading.value == false) {
        print('object');
        return Scaffold(
          appBar: AppBar(
            backgroundColor: backgroundColor2,
            title: const Text("Rekapitulasi Grading"),
          ),
          backgroundColor: backgroundColor1,
          body: ListView(
            children: [
              chartGrading(),
              gradingDataRecap(),
            ],
          ),
        );
      } else {
        return Center(
          child: CircularProgressIndicator(),
        );
      }
    });
  }
}

```

```

        sizingSec(),
        entryGradingButton(),
        recapTitle(),

        controller.list_fishGrading.isEmpty
            ? emptyListGrading()
            : listGrading(),
        SizedBox(
            height: 10,
        )
    ],
),
);
} else {
    return Center(
        child: CircularProgressIndicator(
            color: secondaryColor,
        ),
    );
}
));
}
}
}

```

Code untuk halaman entry grading

```

import 'package:fish/pages/grading/grading_entry_controller.dart';
import 'package:flutter/material.dart';
import 'package:fish/theme.dart';
import 'package:flutter/services.dart';
import 'package:get/get.dart';

import 'grading_controller.dart';

class GradingEntryPage extends StatelessWidget {
    const GradingEntryPage({Key? key}) : super(key: key);

    @override
    Widget build(BuildContext context) {
        final GradingEntryController controller = Get.put(GradingEntryController());
        final GradingController gradingcontroller = Get.put(GradingController());

        Widget fishTypeInput() {...}

        Widget sampleAmountInput() {...}

        Widget fishWightInput() {...}

        Widget fishLengthAvgInput() {...}

        Widget undersizeInput() {...}
    }
}

```

```

Widget oversizeInput() {...}

Widget normalsizeInput() {...}

Widget submitButton() {...}

return Obx(() {
  if (controller.isLoading.value == false) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: backgroundColor2,
        title: Text("Entry Grading ${controller.pond.alias}"),
      ),
      backgroundColor: backgroundColor1,
      body: ListView(
        children: [
          fishTypeInput(),
          sampleAmountInput(),
          fishWightInput(),
          fishLengthAvgInput(),
          normalsizeInput(),
          undersizeInput(),
          oversizeInput(),
          submitButton(),
          SizedBox(
            height: 8,
          )
        ],
      ),
    );
  } else {
    return Center(
      child: CircularProgressIndicator(
        color: secondaryColor,
      ),
    );
  }
});
}

```

Code untuk halaman detail grading

```

import 'package:fish/pages/grading/detail_grading_controller.dart';
import 'package:flutter/material.dart';
import 'package:fish/theme.dart';
import 'package:get/get.dart';

class DetailGradingPage extends StatelessWidget {
  const DetailGradingPage({Key? key}) : super(key: key);

  @override

```

```

Widget build(BuildContext context) {
    final GradingDetailController controller =
        Get.put(GradingDetailController());
    Widget gradingDataRecap() {...}
    Widget detail() {...}
    Widget titleRecap() {...}
    Widget dataGrading() {...}
    Widget detailGrading() {...}

    return Obx(() {
        if (controller.isLoading.value == false) {
            return Scaffold(
                appBar: AppBar(
                    backgroundColor: backgroundColor2,
                    title: const Text("Detail Rekap Grading"),
                ),
                backgroundColor: backgroundColor1,
                body: ListView(
                    children: [
                        gradingDataRecap(),
                        detail(),
                        titleRecap(),
                        dataGrading(),
                        detailGrading(),
                        SizedBox(
                            height: 10,
                        )
                    ],
                ),
            );
        } else {
            return Center(
                child: CircularProgressIndicator(
                    color: secondaryColor,
                ),
            );
        }
    ));
}

```

Membuat model class grading berat ikan

```

import 'package:intl/intl.dart';

class FishGrading {
    String? id;

```

```

String? fishType;
num? samplingAmount;
num? avgFishWeight;
num? avgFishLong;
num? normalFish;
num? oversizeFish;
num? undersizeFish;
DateTime? gradingAt;

FishGrading({
  required this.id,
  required this.fishType,
  required this.samplingAmount,
  required this.avgFishWeight,
  this.avgFishLong,
  this.normalFish,
  this.oversizeFish,
  this.undersizeFish,
  this.gradingAt,
});

factory FishGrading.fromJson(Map<String, dynamic> json) {
  print(json);
  return FishGrading(
    id: json['_id'],
    fishType: json['fish_type'],
    samplingAmount: json['sampling_amount'],
    avgFishWeight: json['avg_fish_weight'],
    avgFishLong: json['avg_fish_long'],
    normalFish: json['amount_normal_fish'],
    oversizeFish: json['amount_oversize_fish'],
    undersizeFish: json['amount_undersize_fish'],
    gradingAt: DateTime.tryParse(json['grading_at']),
  );
}

static List<FishGrading> fromJsonList(List<dynamic> list) {
  List<FishGrading> fishgradings = [];
  for (var item in list) {
    fishgradings.add(FishGrading.fromJson(item));
  }
  return fishgradings;
}

String getDate() => DateFormat('dd-MM-yyyy ').format(gradingAt!);
}

```

Membuat network service untuk fitur grading

```

import 'dart:convert';
import 'package:fish/models/fishGrading_model.dart';
import 'package:fish/models/grading_chart_model.dart';

```

```

import 'package:fish/service/url_api.dart';
import 'package:http/http.dart' as http;

class FishGradingService {
  Future<List<FishGrading>> fetchFishGradings(
    {required String activationId}) async {
    var url = Uri.parse(Urls.fishGrading(activationId));
    var headers = {'Content-Type': 'application/json'};

    var response = await http.get(url, headers: headers);

    print(response.body);

    if (response.statusCode == 200) {
      var data = jsonDecode(response.body);
      List<FishGrading> fishgradings = FishGrading.fromJsonList(data);
      print("success add fishgradings");
      return fishgradings;
    } else {
      throw Exception('Gagal Get fishgradings!');
    }
  }

  Future<List<GradingChartData>> fetchChartFishGradings(
    {required String activationId}) async {
    var url = Uri.parse(Urls.fishGrading(activationId));
    var headers = {'Content-Type': 'application/json'};

    var response = await http.get(url, headers: headers);

    print(response.body);

    if (response.statusCode == 200) {
      var data = jsonDecode(response.body);
      List<GradingChartData> fishgradings = GradingChartData.fromJsonList(data);
      print("success add fishgradings");
      return fishgradings;
    } else {
      throw Exception('Gagal Get fishgradings!');
    }
  }

  Future<bool> postFishGrading({
    required String? pondId,
    required String? fishType,
    required String? samplingAmount,
    required String? avgFishWeight,
    required String? avgFishLong,
    required String? amountNormal,
    required String? amountOver,
    required String? amountUnder,
  }) async {
    if (avgFishWeight!.isNotEmpty) {

```

```

    if (avgFishWeight.contains(","))
        avgFishWeight = avgFishWeight.replaceAll(',', '.');
    }
}
print({
    "pond_id": pondId.toString(),
    "fish_type": fishType,
    "sampling_amount": samplingAmount,
    "avg_fish_weight": avgFishWeight,
    "avg_fish_long": avgFishLong,
    "amount_normal_fish": amountNormal,
    "amount_oversize_fish": amountOver,
    "amount_undersize_fish": amountUnder,
});
final response = await http.post(
    Uri.parse(Urls.fishGradings),
    headers: {
        "Content-Type": "application/x-www-form-urlencoded",
    },
    encoding: Encoding.getByName('utf-8'),
    body: {
        "pond_id": pondId,
        "fish_type": fishType,
        "sampling_amount": samplingAmount,
        "avg_fish_weight": avgFishWeight,
        "avg_fish_long": avgFishLong,
        "amount_normal_fish": amountNormal,
        "amount_oversize_fish": amountOver,
        "amount_undersize_fish": amountUnder,
    },
);
}

if (response.statusCode == 200) {
    print(response.body);
    return true;
} else {
    print(response.body);
    return false;
}
}
}

```

F. Lampiran 6 Code Sprint 5 report

Code halaman rekapitulasi kematian ikan

```

import 'package:fish/pages/component/death_card.dart';
import 'package:fish/pages/fish/fish_recap_controller.dart';
import 'package:flutter/material.dart';
import 'package:fish/pages/fish/fish_death_entry_page.dart';
import 'package:fish/theme.dart';
import 'package:get/get.dart';
import 'package:syncfusion_flutter_charts/charts.dart';

import '../../../../../models/fish_live_model.dart';

class FishRecapPage extends StatelessWidget {
  const FishRecapPage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final FishRecapController controller = Get.put(FishRecapController());

    Widget chartDeath() {...}

    Widget fishDataRecap() {...}

    Widget entryDeathButton() {...}

    Widget emptyListPond() {...}

    Widget detail() {...}

    Widget recapTitle() {...}

    Widget listDeath() {...}

    return Obx(() {
      if (controller.isLoading.value == false) {
        return Scaffold(
          appBar: AppBar(
            backgroundColor: backgroundColor2,
            title: const Text("Rekapitulasi Jumlah Kematian"),
          ),
          backgroundColor: backgroundColor1,
          body: ListView(
            children: [
              chartDeath(),
              fishDataRecap(),
              detail(),
              // sizingSec(),
              entryDeathButton(),
              recapTitle(),
              // chartRecap(),
            ],
          ),
        );
      } else {
        return Center(
          child: CircularProgressIndicator(),
        );
      }
    });
  }
}

```

```

        controller.list_fishDeath.isEmpty ? emptyListPond() : listDeath(),
        SizedBox(
            height: 10,
        )
    ],
),
);
} else {
    return Center(
        child: CircularProgressIndicator(
            color: secondaryColor,
        ),
    );
}
));
}
}
}

```

Code untuk halaman entry kematian

```

import 'package:fish/pages/fish/fish_death_controller.dart';
import 'package:flutter/material.dart';
import 'package:fish/theme.dart';
import 'package:flutter/services.dart';
import 'package:get/get.dart';

import 'fish_recap_controller.dart';

class FishDeathEntryPage extends StatelessWidget {
    const FishDeathEntryPage({Key? key}) : super(key: key);

    @override
    Widget build(BuildContext context) {
        final FishDeathEntryController controller =
            Get.put(FishDeathEntryController());

        final FishRecapController deathcontroller = Get.put(FishRecapController());
        Widget fishTypeInput() {...}

        Widget fishDeathAmountInput() {...}

        Widget submitButton() {...}

        return Obx(() {
            if (controller.isLoading.value == false) {
                return Scaffold(
                    appBar: AppBar(
                        backgroundColor: backgroundColor2,
                        title: Text("Entry Kematian Ikan ${controller.pond.alias}"),
                    ),
                    backgroundColor: backgroundColor1,
                    body: ListView(

```

```

        children: [
            fishTypeInput(),
            fishDeathAmountInput(),
            submitButton(),
            SizedBox(
                height: 8,
            )
        ],
    ),
);
} else {
    return Center(
        child: CircularProgressIndicator(
            color: secondaryColor,
        ),
    );
}
));
}
}

```

Membuat model class rekapitulasi kematian ikan

```

import 'package:intl/intl.dart';

class FishDeath {
    String? id;
    String? diagnosis;
    String? fishType;
    num? deathAmount;
    DateTime? deathAt;

    FishDeath({
        required this.id,
        required this.diagnosis,
        required this.fishType,
        required this.deathAmount,
        required this.deathAt,
    });

    factory FishDeath.fromJson(Map<String, dynamic> json) {
        print(json);
        return FishDeath(
            id: json['_id'],
            diagnosis: json['diagnosis'],
            fishType: json['fish']['fish_type'],
            deathAmount: json["fish"]['fish_amount'] * -1,
            deathAt: DateTime.tryParse(json['death_at']),
        );
    }

    static List<FishDeath> fromJsonList(List<dynamic> list) {

```

```

List<FishDeath> fishDeaths = [];
for (var item in list) {
    fishDeaths.add(FishDeath.fromJson(item));
}
return fishDeaths;
}

String getDate() => DateFormat('dd-MM-yyyy').format(deathAt!);
}

```

Membuat network service untuk fitur rekapitulasi kematian ikan

```

import 'dart:convert';
import 'package:fish/models/fishDeath_model.dart';
import 'package:fish/models/fish_live_model.dart';
import 'package:fish/service/url_api.dart';
import 'package:http/http.dart' as http;

class FishDeathService {
    Future<List<FishDeath>> fetchFishDeaths(
        {required String activationId}) async {
        var url = Uri.parse(Urls.fishDeath(activationId));
        var headers = {'Content-Type': 'application/json'};

        var response = await http.get(url, headers: headers);

        print(response.body);

        if (response.statusCode == 200) {
            var data = jsonDecode(response.body);
            List<FishDeath> fishlive = FishDeath.fromJsonList(data);
            print("success add fishlive");
            return fishlive;
        } else {
            throw Exception('Gagal Get fishdeath!');
        }
    }

    Future<List<FishLiveData>> fetchFishLive(
        {required String activationId}) async {
        var url = Uri.parse(Urls.fishDeath(activationId));
        var headers = {'Content-Type': 'application/json'};

        var response = await http.get(url, headers: headers);

        print(response.body);

        if (response.statusCode == 200) {
            var data = jsonDecode(response.body);
            List<FishLiveData> fishdeath = FishLiveData.fromJsonList(data);
            print("success add fishdeath");
            return fishdeath;
        }
    }
}

```

```
    } else {
        throw Exception('Gagal Get fishdeath!');
    }
}

Future<bool> postFishDeath({
    required String? pondId,
    required List fish,
}) async {
    print({'pond_id': pondId, "fish_death_amount": fish});
    final response = await http.post(
        Uri.parse(Urls.fishDeaths),
        headers: {
            "Content-Type": "application/x-www-form-urlencoded",
        },
        encoding: Encoding.getByName('utf-8'),
        body: {
            "pond_id": pondId,
            "fish_death_amount": fish.toString(),
            "diagnosis": "mati karena sakit"
        },
    );
}

if (response.statusCode == 200) {
    print(response.body);
    return true;
} else {
    print(response.body);
    return false;
}
}
```

G. Lampiran 7 Code Sprint 6 report

Code halaman rekapitulasi treatment

```

import 'package:fish/models/fishDeath_model.dart';
import 'package:fish/pages/component/treatment_card.dart';
import 'package:fish/pages/treatment/treatment_controller.dart';
import 'package:fish/pages/treatment/treatment_entry_page.dart';
import 'package:flutter/material.dart';
import 'package:fish/theme.dart';
import 'package:get/get.dart';

class TreatmentpPage extends StatefulWidget {
    TreatmentpPage({Key? key}) : super(key: key);

    @override
    State<TreatmentpPage> createState() => _TreatmentPageState();
}

class _TreatmentPageState extends State<TreatmentpPage> {
    final TreatmentController controller = Get.put(TreatmentController());

    @override
    void initState() {
        super.initState();
        WidgetsBinding.instance.addPostFrameCallback((timeStamp) async {
            // await controller.getPondActivations(
            //     pondId: controller.pond.id.toString());
            // });
            controller.getTreatmentData(context);
        });
    }

    @override
    void dispose() {
        controller.postDataLog(controller.fitur);
        super.dispose();
    }

    @override
    Widget build(BuildContext context) {...}

    Widget listTreatment() {...}

    Widget emptyListTreatment() {...}

    return Obx(() {
        if (controller.isLoading.value == false) {
            return Scaffold(
                floatingActionButton: FloatingActionButton(
                    onPressed: () {
                        Get.to(() => TreatmentEntryPage(), arguments: {
                            "pond": controller.pond,
                        });
                    },
                ),
            );
        }
    });
}

```

```

        "activation": controller.activation
    });
    controller.postDataLog(controller.fitur);
},
backgroundColor: primaryColor,
child: const Icon(Icons.add),
),
backgroundColor: backgroundColor1,
body: ListView(
children: [
fishDataRecap(),
controller.listTreatmentTest.isEmpty
? emptyListTreatment()
: listTreatment(),
SizedBox(
height: 10,
)
],
),
);
} else {
return Center(
child: CircularProgressIndicator(
color: secondaryColor,
),
);
}
});
}
}

```

Code untuk halaman entry treatment

```

import 'package:fish/models/fish_model.dart';
import 'package:fish/pages/component/treatment_berat_input_card.dart';
import 'package:fish/pages/treatment/treatment_entry_controller.dart';
import 'package:fish/pages/treatment/treatment_controller.dart';
import 'package:flutter/material.dart';
import 'package:fish/pages/pond/detail_pond_controller.dart';
import 'package:fish/theme.dart';

import 'package:fish/pages/component/deactivation_list_input.dart';
import 'package:flutter/services.dart';
import 'package:get/get.dart';

class TreatmentEntryPage extends StatefulWidget {
TreatmentEntryPage({Key? key}) : super(key: key);
@Override
State<TreatmentEntryPage> createState() => _TreatmentEntryPageState();
}

class _TreatmentEntryPageState extends State<TreatmentEntryPage> {
final TreatmentEntryController controller =

```

```

Get.put(TreatmentEntryController());

final TreatmentController treatmentTontroller =
    Get.put(TreatmentController());
void initState() {
    super.initState();
    // WidgetsBinding.instance.addPostFrameCallback((timeStamp) async {
    //     await controller.getPondActivations(
    //         pondId: controller.pond.id.toString());
    // });
    controller.getHarvestedBool(controller.activation);
}

@Override
void dispose() {
    controller.postDataLog(controller.fitur);
    controller.carbonController.clear();
    controller.descController.clear();
    controller.leleWeightController.clear();
    controller.masWeightController.clear();
    controller.nilaHitamWeightController.clear();
    controller.nilaMerahWeightController.clear();
    controller.patinWeightController.clear();
    controller.saltController.clear();
    controller.waterController.clear();
    controller.probioticController.clear();
    super.dispose();
}

@Override
Widget build(BuildContext context) {
    Widget descInput() {...}

    Widget carbonTypeNullInput() {...}

    Widget waterChangeInput() {...}

    Widget listTreatmentBeratInput() {...}

    Widget probioticInput() {...}

    Widget carbonInput() {...}

    Widget carbonTypeInput() {...}

    Widget submitButton() {...}

    Widget submitBeratButton() {...}

    return Obx(() {
        if (controller.isLoading.value == false) {
            return Scaffold(
                appBar: AppBar(

```

```

        backgroundColor: backgroundColor2,
        title: Text("Entry Treatment Kolam ${controller.pond.alias}"),
      ),
      backgroundColor: backgroundColor1,
      body: ListView(
        children: [
          treatmentTypeInput(),
          descInput(),
          controller.typeController.selected.value == "berat"
            ? listTreatmentBeratInput()
            : waterChangeInput(),
          controller.typeController.selected.value == "berat"
            ? Container()
            : probioticInput(),
          controller.typeController.selected.value == "berat"
            ? Container()
            : carbonTypeInput(),
          controller.carbonTypeController.selected.value == "tidak ada"
            ? Container()
            : carbonInput(),
          controller.typeController.selected.value == "berat"
            ? Container()
            : saltDosisInput(),
          controller.typeController.selected.value == "berat"
            ? submitBeratButton()
            : submitButton(),
          SizedBox(
            height: 8,
          )
        ],
      ),
    );
  } else {
    return Center(
      child: CircularProgressIndicator(
        color: secondaryColor,
      ),
    );
  }
);
}
}

```

Code untuk halaman detail treatment

```

import 'package:fish/pages/treatment/treatment_detail_controller.dart';
import 'package:flutter/material.dart';
import 'package:fish/theme.dart';
import 'package:get/get.dart';

class DetailTreatmentPage extends StatelessWidget {
  const DetailTreatmentPage({Key? key}) : super(key: key);

```

```
@override
Widget build(BuildContext context) {
    final TreatmentDetailController controller =
        Get.put(TreatmentDetailController());

    Widget treatmentDataRecap() {...}

    Widget detail() {...}

    Widget titleRecap() {...}

    Widget dataTreatment() {...}

    Widget detailTreatment() {...}

    return Obx(() {
        if (controller.isLoading.value == false) {
            return Scaffold(
                appBar: AppBar(
                    backgroundColor: backgroundColor2,
                    title: const Text("Detail Treatment "),
                ),
                backgroundColor: backgroundColor1,
                body: ListView(
                    children: [
                        treatmentDataRecap(),
                        detail(),
                        titleRecap(),
                        dataTreatment(),
                        detailTreatment(),
                        SizedBox(
                            height: 10,
                        )
                    ],
                ),
            );
        } else {
            return Center(
                child: CircularProgressIndicator(
                    color: secondaryColor,
                ),
            );
        }
    });
}
```

H. Lampiran 8 Code Sprint 7 report

Membuat model class fitur treatment kolam

```

import 'dart:ffi';

import 'package:intl/intl.dart';

class Treatment {
    String? id;
    String? activation_id;
    num? salt;
    String? type;
    num? probiotic;
    num? water;
    num? carbohydrate;
    String? carbohydrate_type;
    String? desc;
    String? treatmentAt;

    Treatment(
        {this.id,
        this.salt,
        this.type,
        this.probiotic,
        this.water,
        this.carbohydrate,
        this.desc,
        this.carbohydrate_type,
        this.activation_id,
        this.treatmentAt});

    // Treatment.fromJson(Map<String, dynamic> json) {
    //   salt = json['salt'];
    //   type = json['treatment_type'];
    //   probiotic = json['probiotic_culture'];
    // }
    factory Treatment.fromJson(Map<String, dynamic> json) {
        return Treatment(
            id: json['id'],
            salt: json['salt'],
            type: json['treatment_type'],
            probiotic: json['probiotic_culture'],
            water: json['water_change'],
            desc: json['description'],
            carbohydrate: json['carbohydrate'],
            carbohydrate_type: json['carbohydrate_type'],
            activation_id: json['pond_activation_id'],
            treatmentAt: json["treatment_at"]);
    }
    String getGmtToNormalDate() {
        String stringDate = treatmentAt!;

```

```

        DateTime dateTime = DateFormat("yyyy-MM-dd hh:mm:ss").parse(stringDate);
        String newStringDate = DateFormat("dd-MM-yyyy").format(dateTime);
        return newStringDate;
    }
}

```

Membuat network service untuk fitur treatment kolam

```

import 'dart:convert';
import 'package:fish/models/treatment_model.dart';
import 'package:fish/service/url_api.dart';
import 'package:http/http.dart' as http;

class TreatmentService {
    Future<List<Treatment>> getTreatmentList() async {
        var url = Uri.parse(Urls.treatment);
        var headers = {'Content-Type': 'application/json'};

        var response = await http.get(url, headers: headers);

        print(response.body);

        if (response.statusCode == 200) {
            var data = jsonDecode(response.body);
            List<Treatment> treatments = [];

            for (var item in data) {
                treatments.add(Treatment.fromJson(item));
            }
            // Treatment treatment = Treatment.fromJson(data[0]);
            // print(data[1]);
            return treatments;
        } else {
            throw Exception('Gagal Get Products!');
        }
    }

    Future<bool> postPondTreatment({
        required String? pondId,
        String? salt,
        String? type,
        String? probiotic,
        String? water,
        String? desc,
        String? carbohydrate,
        String? carbohydrate_type,
    }) async {
        print({
            "pond_id": pondId.toString(),
            "salt": salt,
            "treatment_type": type,
            "probiotic_culture": probiotic,
        });
    }
}

```

```

    "water_change": water,
    "description": desc,
    "carbohydrate": carbohydrate,
    "carbohydrate_type": carbohydrate_type,
  });
final response = await http.post(
  Uri.parse(Urls.treatment),
  headers: {
    "Content-Type": "application/x-www-form-urlencoded",
  },
  encoding: Encoding.getByName('utf-8'),
  body: {
    "pond_id": pondId,
    "salt": salt,
    "treatment_type": type,
    "probiotic_culture": probiotic,
    "water_change": water,
    "description": desc.toString(),
    "carbohydrate": carbohydrate,
    "carbohydrate_type": carbohydrate_type,
  },
);
if (response.statusCode == 200) {
  print(response.body);
  return true;
} else {
  print(response.body);
  return false;
}
}

Future<bool> postPondTreatmentBerat(
  required String? pondId,
  String? type,
  String? desc,
  required num? total_fish_harvested,
  required num? total_weight_harvested,
  List? fish_harvested,
  bool? isFinish)) async {
print({
  "pond_id": pondId.toString(),
  "treatment_type": type,
  "description": desc,
});
final response = await http.post(
  Uri.parse(Urls.treatment),
  headers: {
    "Content-Type": "application/x-www-form-urlencoded",
  },
  encoding: Encoding.getByName('utf-8'),
  body: {
    "pond_id": pondId,
  }
);
if (response.statusCode == 200) {
  print(response.body);
  return true;
} else {
  print(response.body);
  return false;
}
}
}

```

```
    "treatment_type": type,
    "description": desc.toString(),
    "total_weight_harvested": total_weight_harvested.toString(),
    "total_fish_harvested": total_fish_harvested.toString(),
    "fish": fish_harvested.toString()
  },
);

if (response.statusCode == 200) {
  print(response.body);
  return true;
} else {
  print(response.body);
  return false;
}
}
```

I. Lampiran 9 Code Sprint 8 report

Code halaman list pencatatan kualitas air harian

```

import
  'package:fish/controllers/daily_water/daily_water_breed_list_controller.dart';
import 'package:fish/pages/component/daily_water_card.dart';
import 'package:fish/controllers/daily_water/daily_water_controller.dart';
import 'package:fish/pages/dailywater/daily_water_entry_page.dart';
import 'package:fish/pages/pond/pond_controller.dart';
import 'package:flutter/material.dart';
import 'package:fish/theme.dart';
import 'package:get/get.dart';
import 'package:http/http.dart';

import 'daily_water_avg.dart';

class DailyWaterPage extends StatefulWidget {
  DailyWaterPage({Key? key}) : super(key: key);

  @override
  State<DailyWaterPage> createState() => _DailyWaterPageState();
}

class _DailyWaterPageState extends State<DailyWaterPage> {
  final DailyWaterController controller = Get.put(DailyWaterController());
  final PondController pondController = Get.find();
  final DailyWaterBreedListController dailyWaterBreedListController =
    Get.find();

  @override
  void initState() {
    super.initState();
    // WidgetsBinding.instance.addPostFrameCallback((timeStamp) async {
    //   await controller.getPondActivations(
    //     pondId: controller.pond.id.toString(),
    //   );
    // });
    controller.getDailyWaterData(context);
    controller.startTime = DateTime.now();
    print('ini init state');
  }

  @override
  void dispose() {
    controller.postDataLog(controller.fitur);
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    controller.startTime = DateTime.now();
    print('ini build daily water');
  }
}

```

```

Widget fishDataRecap() {...}

Widget listDailyWater() {...}

Widget emptyList() {...}

return Obx(() {
  if (controller.isLoading.value == false) {
    return Scaffold(
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          Get.to(() => DailyWaterEntryPage(), arguments: {
            "pond": pondController.selectedPond.value,
            "activation": dailyWaterBreedListController.selectedActivation.value
          });
          controller.postDataLog(controller.fitur);
        },
        backgroundColor: primaryColor,
        child: const Icon(Icons.add),
      ),
      backgroundColor: backgroundColor1,
      body: ListView(
        children: [
          fishDataRecap(),
          controller.listDailyWater.isEmpty
              ? emptyList()
              : listDailyWater(),
          SizedBox(
            height: 10,
          )
        ],
      ),
    );
  } else {
    return Center(
      child: CircularProgressIndicator(
        color: secondaryColor,
      ),
    );
  }
));
}

```

Code untuk halaman entry kualitas air harian

```

import 'package:fish/controllers/daily_water/daily_water_entry_controller.dart';
import 'package:fish/controllers/daily_water/daily_water_controller.dart';
import 'package:flutter/material.dart';
import 'package:fish/theme.dart';
import 'package:flutter/services.dart';

```

```
import 'package:get/get.dart';

import '../component/tabviewwater.dart';

class DailyWaterEntryPage extends StatelessWidget {
    DailyWaterEntryPage({Key? key}) : super(key: key);

    final DailyWaterEntryController controller =
        Get.put(DailyWaterEntryController());

    final DailyWaterController water = Get.put(DailyWaterController());

    @override
    Widget build(BuildContext context) {
        Widget doInput() {...}

        Widget phInput() {...}

        Widget temperatureInput() {...}

        Widget submitButton() {...}

        return Obx(() {
            if (controller.isLoading.value == false) {
                return Scaffold(
                    appBar: AppBar(),
                    backgroundColor: backgroundColor2,
                    title:
                        Text("Entry Kondisi Air Harian Kolam ${controller.pond.alias}"),
                    ),
                    backgroundColor: backgroundColor1,
                    body: ListView(
                    children: [
                        phInput(),
                        doInput(),
                        temperatureInput(),
                        submitButton(),
                        SizedBox(
                        height: 8,
                        )
                    ],
                    ),
                );
            } else {
                return Center(
                    child: CircularProgressIndicator(
                    color: secondaryColor,
                    ),
                );
            }
        });
    }
}
```

Code untuk halaman detail kualitas air harian

```

import
    'package:fish/controllers/daily_water/daily_water_breed_list_controller.dart';
import
    'package:fish/controllers/daily_water/daily_water_detail_controller.dart';
import 'package:fish/pages/dailywater/daily_water_edit_page.dart';
import 'package:fish/pages/pond/detail_pond_controller.dart';
import 'package:fish/pages/pond/pond_controller.dart';
import 'package:flutter/material.dart';
import 'package:fish/theme.dart';
import 'package:flutter/services.dart';
import 'package:get/get.dart';

import '../../../../../controllers/daily_water/daily_water_controller.dart';
import '../../../../../controllers/daily_water/daily_water_edit_controller.dart';
import '../../../../../models/daily_water_model.dart';

class DailyWaterDetailPage extends StatefulWidget {
    const DailyWaterDetailPage({Key? key}) : super(key: key);

    @override
    State<DailyWaterDetailPage> createState() => _DailyWaterDetailPageState();
}

@override
class _DailyWaterDetailPageState extends State<DailyWaterDetailPage> {
    final DailyWaterDetailController controller =
        Get.put(DailyWaterDetailController());
    final PondController pondController = Get.find();
    final DailyWaterBreedListController dailyWaterBreedListController =
        Get.find();
    final DailyWaterController dailyWaterController = Get.find();
    final DailyWaterEditController editController =
        Get.put(DailyWaterEditController());
    @override
    void initState() {
        super.initState();
        controller.getDailyWaterData(
            context, dailyWaterController.selectedDailyWater.value.id.toString());
    }

    @override
    void dispose() {
        controller.postDataLog(controller.fitur);
        dailyWaterController.getDailyWaterData(context);
        super.dispose();
    }

    @override
    Widget build(BuildContext context) {
        Widget airHarianDataRecap() {...}

```

```

Widget detail() {...}

Widget titleRecap() {...}

Widget dataAirHarian() {...}

Widget detailAirHarian() {...}

return Obx(() {
  if (controller.isLoading.value == false) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: backgroundColor2,
        title: const Text("Detail Kondisi Air Harian"),
        leading: new IconButton(
          icon: new Icon(Icons.arrow_back),
          onPressed: () async {
            // Get.back();
            Navigator.pop(context);
          },
        ),
        backgroundColor: backgroundColor1,
        body: ListView(
          children: [
            airHarianDataRecap(),
            detail(),
            titleRecap(),
            dataAirHarian(),
            detailAirHarian(),
            SizedBox(
              height: 10,
            )
          ],
        ),
      );
  } else {
    return Center(
      child: CircularProgressIndicator(
        color: secondaryColor,
      ),
    );
  }
));
}

void editDataPh(DailyWater dailywater, String title) {
  showDialog<String>(
    context: context,
    builder: (BuildContext context) => AlertDialog(
      title: Text('Edit $title', style: TextStyle(color: Colors.white)),
      content: Container(

```

```

height: 50,
padding: EdgeInsets.symmetric(
    horizontal: 16,
),
decoration: BoxDecoration(
    color: backgroundColor2,
    borderRadius: BorderRadius.circular(12),
),
child: Center(
    child: TextFormField(
        style: primaryTextStyle,
        inputFormatters: <TextInputFormatter>[
            FilteringTextInputFormatter.deny(RegExp(r'[-+=*#%/, \s]+'))
        ],
        keyboardType: TextInputType.number,
        controller: controller.phController,
    )),
),
backgroundColor: backgroundColor1,
actions: <Widget>[
    TextButton(
        onPressed: () => Navigator.pop(context, 'Batal'),
        child: const Text(
            'Batal',
            style: TextStyle(color: Colors.red),
        ),
    ),
    TextButton(
        onPressed: () async {
            await editController.editDailyWaterDataOne(context, () {
                Navigator.pop(context, 'Submit');
            },
            controller.phController.text,
            dailywater.numDo.toString(),
            dailywater.temperature.toString());
            controller.getDailyWaterData(
                context,
                dailyWaterController.selectedDailyWater.value.id
                    .toString());
        },
        child: const Text('Submit'),
    ),
],
));
}

void editDataSuhu(DailyWater dailywater, String title) {
showDialog<String>(
    context: context,
    builder: (BuildContext context) => AlertDialog(
        title: Text('Edit $title', style: TextStyle(color: Colors.white)),
        content: Container(
            height: 50,

```



```

decoration: BoxDecoration(
  color: backgroundColor2,
  borderRadius: BorderRadius.circular(12),
),
child: Center(
  child: TextFormField(
    style: primaryTextStyle,
    inputFormatters: <TextInputFormatter>[
      FilteringTextInputFormatter.deny(RegExp(r'[-+=*#%/, \s]+'))
    ],
    keyboardType: TextInputType.number,
    controller: controller.doController,
  )),
),
backgroundColor: backgroundColor1,
actions: <Widget>[
  TextButton(
    onPressed: () => Navigator.pop(context, 'Batal'),
    child: const Text(
      'Batal',
      style: TextStyle(color: Colors.red),
    ),
  ),
  TextButton(
    onPressed: () async {
      await editController.editDailyWaterDataOne(context, () {
        Navigator.pop(context, 'Submit');
      }, dailywater.ph.toString(), controller.doController.text,
        dailywater.numDo.toString());
      controller.getDailyWaterData(
        context,
        dailyWaterController.selectedDailyWater.value.id
          .toString());
    },
    child: const Text('Submit'),
  ),
],
));
}
}

```

Membuat model class fitur pencatatan kualitas air harian

```

import 'package:flutter/material.dart';
import 'package:intl/intl.dart';

class DailyWater {
  String? id;
  String? pondId;
  String? activationId;
  num? ph;

```

```

num? numDo;
num? temperature;
num? week;
String? ph_desc;
String? numDo_desc;
String? dailywater_at;

DailyWater(
    required this.id,
    this.pondId,
    this.activationId,
    this.numDo,
    this.numDo_desc,
    this.temperature,
    this.week,
    this.ph,
    this.ph_desc,
    this.dailywater_at));

factory DailyWater.fromJson(Map<String, dynamic> json) {
    return DailyWater(
        id: json['_id'],
        pondId: json['pond_id'],
        activationId: json['pond_activation_id'],
        ph: json['ph'],
        numDo: json['do'],
        temperature: json['temperature'],
        week: json['week'],
        ph_desc: json['ph_desc'],
        numDo_desc: json['do_desc'],
        dailywater_at: json['dailywater_at']);
}

String getGmtToNormalDate() {
    String stringDate = dailywater_at!;
    DateTime dateTime = DateFormat("yyyy-MM-dd hh:mm:ss").parse(stringDate);
    String newStringDate = DateFormat("dd-MM-yyyy").format(dateTime);
    return newStringDate;
}

String getDayNameDate() {
    String stringDate = dailywater_at!;
    DateTime dateTime = DateFormat("yyyy-MM-dd hh:mm:ss").parse(stringDate);
    String newStringDate = DateFormat("EEEE").format(dateTime);
    return newStringDate;
}
}

```

Membuat network service untuk fitur pencatatan kualitas air harian

```

import 'dart:convert';
import 'package:fish/models/daily_water_model.dart';

```

```

import 'package:fish/service/url_api.dart';
import 'package:http/http.dart' as http;

class DailyWaterService {
  Future<List<DailyWater>> getPonds() async {
    var url = Uri.parse(Urls.dailyWater);
    var headers = {'Content-Type': 'application/json'};

    var response = await http.get(url, headers: headers);

    print(response.body);

    if (response.statusCode == 200) {
      var data = jsonDecode(response.body);
      List<DailyWater> ponds = [];

      for (var item in data) {
        ponds.add(DailyWater.fromJson(item));
      }

      print(ponds);

      return ponds;
    } else {
      throw Exception('Gagal Get Ponds!');
    }
  }

  Future<bool> postDailyWater({
    required String? pondId,
    required String? activationId,
    required String? ph,
    required String? numDo,
    String? week,
    required String? temperature,
  }) async {
    print({
      "pond_id": pondId.toString(),
      "pond_activation_id": activationId.toString(),
      "ph": ph,
      "do": numDo,
      "week": week,
      "temperature": temperature,
    });
    final response = await http.post(
      Uri.parse(Urls.dailyWater),
      headers: {
        "Content-Type": "application/x-www-form-urlencoded",
      },
      encoding: Encoding.forName('utf-8'),
      body: {
        "pond_id": pondId.toString(),
        "pond_activation_id": activationId.toString(),
      }
    );
  }
}

```

```

        "ph": ph,
        "week": week,
        "do": numDo,
        "temperature": temperature,
    },
),
);

if (response.statusCode == 200) {
    print(response.body);
    return true;
} else {
    print(response.body);
    return false;
}
}

Future<bool> editDailyWater(
    required String? dailywaterId,
    required String? ph,
    required String? numDo,
    required String? temperature)) async {
print({
    "ph": ph,
    "do": numDo,
    "temperature": temperature,
});
final response = await http.put(
    Uri.parseUrls.dailyWaterbyid(dailywaterId)),
headers: {
    "Content-Type": "application/x-www-form-urlencoded",
},
encoding: Encoding.getByName('utf-8'),
body: {
    "ph": ph,
    "do": numDo,
    "temperature": temperature,
},
);
}

if (response.statusCode == 200) {
    print(response.body);
    return true;
} else {
    print(response.body);
    return false;
}
}

Future<DailyWater> DeleteDatas({required String dailywaterId}) async {
var url = Uri.parseUrls.dailyWaterbyid(dailywaterId));
var headers = {'Content-Type': 'application/json'};

var response = await http.delete(url, headers: headers);

```

```
print(response.body);

if (response.statusCode == 200) {
    return DailyWater.fromJson(jsonDecode(response.body));
} else {
    throw Exception('Gagal Get Ponds!');
}
}

Future<List<DailyWater>> getDatas({required String dailywaterId}) async {
var url = Uri.parse(Urls.dailyWaterbyid(dailywaterId));
var headers = {'Content-Type': 'application/json'};

var response = await http.get(url, headers: headers);

print(response.body);

if (response.statusCode == 200) {
    var data = jsonDecode(response.body);
    List<DailyWater> ponds = [];
    ponds.add(DailyWater.fromJson(data));

    print(ponds);

    return ponds;
} else {
    throw Exception('Gagal Get Ponds!');
}
}
```

J. Lampiran 10 Code Sprint 9 report

Code halaman list pencatatan kualitas air mingguan

```

import 'package:fish/pages/component/daily_water_card.dart';
import 'package:fish/controllers/weeklywater/weekly_water_controller.dart';
import 'package:fish/pages/component/weekly_water_card.dart';
import 'package:fish/pages/weeklywater/weeklywater_avg.dart';
import 'package:fish/pages/weeklywater/weeklywater_entry_page.dart';
import 'package:flutter/material.dart';
import 'package:fish/theme.dart';
import 'package:get/get.dart';

class WeeklyWaterPage extends StatefulWidget {
    WeeklyWaterPage({Key? key}) : super(key: key);

    @override
    State<WeeklyWaterPage> createState() => _WeeklyWaterPageState();
}

class _WeeklyWaterPageState extends State<WeeklyWaterPage> {
    final WeeklyWaterController controller = Get.put(WeeklyWaterController());

    @override
    void initState() {
        super.initState();
        controller.getWeeklyWaterData(context);
    }

    @override
    void dispose() {
        controller.postDataLog(controller.fitur);
        super.dispose();
    }

    @override
    Widget build(BuildContext context) {
        Widget fishDataRecap() {...}

        Widget listWeeklyWater() {...}

        Widget emptyList() {...}

        return Obx(() {
            if (controller.isLoading.value == false) {
                return Scaffold(
                    floatingActionButton: FloatingActionButton(
                        onPressed: () {
                            Get.to(() => WeeklyWaterEntryPage(), arguments: {
                                "pond": controller.pond,
                                "activation": controller.activation
                            });
                        }
                );
            }
        });
    }
}

```

```

        controller.postDataLog(controller.fitur);
    },
    backgroundColor: primaryColor,
    child: const Icon(Icons.add),
),
backgroundColor: backgroundColor1,
body: ListView(
children: [
    fishDataRecap(),
    controller.listWeeklyWater.isEmpty
        ? emptyList()
        : listWeeklyWater(),
    SizedBox(
height: 10,
)
],
),
),
);
} else {
return Center(
    child: CircularProgressIndicator(
    color: secondaryColor,
),
);
}
));
}
}
}

```

Code untuk halaman entry kualitas air mingguan

```

import
    'package:fish/controllers/weeklywater/weekly_water_entry_controller.dart';
import 'package:fish/controllers/weeklywater/weekly_water_controller.dart';
import 'package:flutter/material.dart';
import 'package:fish/theme.dart';
import 'package:flutter/services.dart';
import 'package:get/get.dart';

class WeeklyWaterEntryPage extends StatelessWidget {
    WeeklyWaterEntryPage({Key? key}) : super(key: key);

    final WeeklyWaterEntryController controller =
        Get.put(WeeklyWaterEntryController());

    final WeeklyWaterController weeklyWaterController =
        Get.put(WeeklyWaterController());

    @override
    Widget build(BuildContext context) {
        Widget amoniaInput() {...}

```

```
Widget flocInput() {...}

Widget nitriteInput() {...}

Widget nitrateInput() {...}

Widget hardnessInput() {...}

Widget submitButton() {...}

return Obx(() {
  if (controller.isLoading.value == false) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: backgroundColor2,
        title: const Text("Entry Kondisi Air Harian"),
      ),
      backgroundColor: backgroundColor1,
      body: ListView(
        children: [
          flocInput(),
          amoniaInput(),
          nitriteInput(),
          nitrateInput(),
          hardnessInput(),
          submitButton(),
          SizedBox(
            height: 8,
          )
        ],
      ),
    );
  } else {
    return Center(
      child: CircularProgressIndicator(
        color: secondaryColor,
      ),
    );
  }
});
```

Code untuk halaman detail kualitas air mingguan

```
import 'package:fish/controllers/weeklywater/weekly_water_detail_controller.dart';
import 'package:flutter/material.dart';
import 'package:fish/theme.dart';
import 'package:get/get.dart';

class WeeklyWaterDetailPage extends StatelessWidget {
```

```

const WeeklyWaterDetailPage({Key? key}) : super(key: key);

@Override
Widget build(BuildContext context) {
    final WeeklyWaterDetailController controller =
        Get.put(WeeklyWaterDetailController());

    Widget airMingguanDataRecap() {...}

    Widget detail() {...}

    Widget titleRecap() {...}

    Widget dataAirMingguan() {...}

    Widget detailAirMingguan() {...}

    return Obx(() {
        if (controller.isLoading.value == false) {
            return Scaffold(
                appBar: AppBar(
                    backgroundColor: backgroundColor2,
                    title: const Text("Detail Kondisi Air Harian"),
                ),
                backgroundColor: backgroundColor1,
                body: ListView(
                    children: [
                        airMingguanDataRecap(),
                        detail(),
                        titleRecap(),
                        dataAirMingguan(),
                        detailAirMingguan(),
                        SizedBox(
                            height: 10,
                        )
                    ],
                ),
            );
        } else {
            return Center(
                child: CircularProgressIndicator(
                    color: secondaryColor,
                ),
            );
        }
    )));
}

```

Membuat model class fitur pencatatan kualitas air mingguan

```
import 'package:flutter/material.dart';
```

```
import 'package:intl/intl.dart';

class WeeklyWater {
    String? id;
    String? pondId;
    String? activationId;
    num? ammonia;
    num? floc;
    num? nitrite;
    num? nitrate;
    num? hardness;
    num? week;
    String? floc_desc;
    String? ammonia_desc;
    String? hardness_desc;
    String? nitrate_desc;
    String? nitrite_desc;
    String? weeklywater_at;

    WeeklyWater(
        {required this.id,
        this.pondId,
        this.activationId,
        this.floc,
        this.floc_desc,
        this.ammonia,
        this.ammonia_desc,
        this.nitrate,
        this.nitrate_desc,
        this.nitrite,
        this.nitrite_desc,
        this.hardness,
        this.hardness_desc,
        this.week,
        this.weeklywater_at});

    factory WeeklyWater.fromJson(Map<String, dynamic> json) {
        return WeeklyWater(
            id: json['_id'],
            pondId: json['pond_id'],
            activationId: json['pond_activation_id'],
            floc: json['floc'],
            nitrate: json['nitrate'],
            nitrite: json['nitrite'],
            ammonia: json['ammonia'],
            hardness: json['hardness'],
            floc_desc: json['floc_desc'],
            nitrate_desc: json['nitrate_desc'],
            nitrite_desc: json['nitrite_desc'],
            ammonia_desc: json['ammonia_desc'],
            hardness_desc: json['hardness_desc'],
            week: json['week'],
            weeklywater_at: json['weeklywater_at']);
    }
}
```

```

    }
    String getGmtToNormalDate() {
        String stringDate = weeklywater_at!;
        DateTime dateTime = DateFormat("yyyy-MM-dd hh:mm:ss").parse(stringDate);
        String newStringDate = DateFormat("dd-MM-yyyy").format(dateTime);
        return newStringDate;
    }

    String getDayNameDate() {
        String stringDate = weeklywater_at!;
        DateTime dateTime = DateFormat("yyyy-MM-dd hh:mm:ss").parse(stringDate);
        String newStringDate = DateFormat("EEEE").format(dateTime);
        return newStringDate;
    }
}

```

Membuat network service untuk fitur pencatatan kualitas air mingguan

```

import 'dart:convert';
import 'package:fish/models/weeklywater_model.dart';
import 'package:fish/service/url_api.dart';
import 'package:http/http.dart' as http;

class WeeklyWaterService {
    Future<List<WeeklyWater>> getDatas() async {
        var url = Uri.parse(Urls.weeklyWater);
        var headers = {'Content-Type': 'application/json'};

        var response = await http.get(url, headers: headers);

        print(response.body);

        if (response.statusCode == 200) {
            var data = jsonDecode(response.body);
            List<WeeklyWater> ponds = [];

            for (var item in data) {
                ponds.add(WeeklyWater.fromJson(item));
            }

            print(ponds);

            return ponds;
        } else {
            throw Exception('Gagal Get Ponds!');
        }
    }

    Future<bool> postWeeklyWater({
        required String? pondId,
        required String? activationId,
        required String? floc,
    })
}

```

```
String? ammonia,
String? nitrite,
String? nitrate,
String? hardness,
String? week,
}) async {
print({
  "pond_id": pondId.toString(),
  "pond_activation_id": activationId.toString(),
  "floc",
  "nitrite": nitrate,
  "nitrate": nitrate,
  "ammonia": ammonia,
  "hardness": hardness,
  "week": week,
});
final response = await http.post(
  Uri.parse(Urls.weeklyWater),
  headers: {
    "Content-Type": "application/x-www-form-urlencoded",
  },
  encoding: Encoding.getByName('utf-8'),
  body: {
    "pond_id": pondId.toString(),
    "pond_activation_id": activationId.toString(),
    "floc",
    "nitrite": nitrate,
    "nitrate": nitrate,
    "ammonia": ammonia,
    "hardness": hardness,
    "week": week,
  },
);
if (response.statusCode == 200) {
  print(response.body);
  return true;
} else {
  print(response.body);
  return false;
}
}
```

K. Lampiran 11 Code Sprint 10 report

Code halaman rekapitulasi sortir

```

import 'package:fish/pages/component/transfer_card.dart';
import 'package:fish/controllers/fish_transfer/fish_transfer_list_controller.dart';
import 'package:fish/pages/fish_transfer/fish_transfer_entry_page.dart';
import 'package:flutter/material.dart';
import 'package:fish/theme.dart';
import 'package:get/get.dart';

import 'new_fish_transfer_entry_page.dart';

class FishTransferListPage extends StatefulWidget {
    FishTransferListPage({Key? key}) : super(key: key);

    @override
    State<FishTransferListPage> createState() => _FishTransferListPageState();
}

class _FishTransferListPageState extends State<FishTransferListPage> {
    final TransferController controller = Get.put(TransferController());

    @override
    void initState() {
        super.initState();

        controller.getTransfertData(context);
    }

    @override
    void dispose() {
        controller.postDataLog(controller.fitur);
        super.dispose();
    }

    @override
    Widget build(BuildContext context) {...}

    Widget listTransfer() {...}

    Widget emptyListTransfer() {...}

    return Obx(() {
        if (controller.isLoading.value == false) {
            return Scaffold(
                floatingActionButton: FloatingActionButton(
                    onPressed: () {
                        Get.to(() => const NewFishTransferEntryPage(), arguments: {
                            "pond": controller.pond,
                            "activation": controller.activation
                        });
                    }
                )
            );
        } else {
            return Center(
                child: CircularProgressIndicator()
            );
        }
    });
}

```

```

        controller.postDataLog(controller.fitur);
    },
    backgroundColor: primaryColor,
    child: const Icon(Icons.add),
),
backgroundColor: backgroundColor1,
body: ListView(
children: [
    fishDataRecap(),
    controller.listTransfer.isEmpty
        ? emptyListTransfer()
        : listTransfer(),
    SizedBox(
height: 10,
)
],
),
),
);
} else {
return Center(
    child: CircularProgressIndicator(
color: secondaryColor,
),
);
}
));
}
}

```

Code untuk halaman entry sortir

```

import 'package:fish/models/fish_model.dart';
import 'package:fish/pages/component/treatment_berat_input_card.dart';
import 'package:fish/pages/treatment/treatment_entry_controller.dart';
import
    'package:fish/controllers/fish_transfer/fish_transfer_entry_controller.dart';
import
    'package:fish/controllers/fish_transfer/pond_list_item_controller.dart';
import 'package:flutter/material.dart';
import 'package:fish/pages/pond/detail_pond_controller.dart';
import 'package:fish/theme.dart';

import 'package:fish/pages/component/deactivation_list_input.dart';
import 'package:flutter/services.dart';
import 'package:get/get.dart';

import '../../../../../controllers/fish_transfer/fish_transfer_list_controller.dart';
import '../component/deactivation_with_fish_transfer_input.dart';
import '../component/fish_transfer_input.dart';

class FishTransferEntryPage extends StatefulWidget {
    FishTransferEntryPage({Key? key}) : super(key: key);
    @override

```

```

        State<FishTransferEntryPage> createState() => _FishTransferEntryPageState();
    }

    class _FishTransferEntryPageState extends State<FishTransferEntryPage> {
        final FishTransferEntryController controller =
            Get.put(FishTransferEntryController());

        final TransferController fishTransferController =
            Get.put(TransferController());

        final PondListController getpondlistcontroller =
            Get.put(PondListController());

        final pageController = PageController(initialPage: 0);
        @override
        void dispose() {
            controller.descController.clear();
            controller.sampleLongController.clear();
            controller.sampleWeightController.clear();
            controller.leleAmountActivationController.clear();
            controller.leleAmountController.clear();
            controller.leleAmountDeactivationController.clear();
            controller.leleWeightActivationController.clear();
            controller.leleWeightController.clear();
            controller.leleWeightDeactivationController.clear();
            controller.masAmountActivationController.clear();
            controller.masAmountController.clear();
            controller.masAmountDeactivationController.clear();
            controller.masWeightActivationController.clear();
            controller.masWeightController.clear();
            controller.masWeightDeactivationController.clear();
            controller.patinAmountActivationController.clear();
            controller.patinAmountController.clear();
            controller.patinWeightActivationController.clear();
            controller.patinWeightController.clear();
            controller.patinWeightDeactivationController.clear();
            controller.nilaMerahAmountActivationController.clear();
            controller.nilaMerahAmountController.clear();
            controller.nilaMerahAmountDeactivationController.clear();
            controller.nilaMerahWeightActivationController.clear();
            controller.nilaMerahWeightController.clear();
            controller.nilaMerahWeightDeactivationController.clear();
            controller.nilaHitamAmountActivationController.clear();
            controller.nilaHitamAmountController.clear();
            controller.nilaHitamAmountDeactivationController.clear();
            controller.nilaHitamWeightActivationController.clear();
            controller.nilaHitamWeightController.clear();
            controller.nilaHitamWeightDeactivationController.clear();
            pageController.dispose();
            controller.postDataLog(controller.fitur);
            super.dispose();
        }
    }
}

```

```
void initState() {
    super.initState();
    // WidgetsBinding.instance.addPostFrameCallback((timeStamp) async {
    //   await controller.getPondActivations(
    //     pondId: controller.pond.id.toString());
    // });
    // controller.getHarvestedBool(controller.activation);
    controller.getPondsData(controller.methodController.toString());
    controller.getHarvestedBool(controller.activation);
}

@Override
Widget build(BuildContext context) {
    Widget sampleWeightInput() {...}

    Widget sampleLongInput() {...}

    Widget destinationPondInput() {...}

    Widget transferMethodInput() {...}

    Widget checkBoxFishTransfer() {...}

    Widget leleInput() {...}

    Widget nilaMerahInput() {...}

    Widget nilaHitamInput() {...}

    Widget patinInput() {...}

    Widget masInput() {...}

//input aktivasi
    Widget checkBoxFish() {...}

    Widget waterHeightInput() {...}

    Widget leleInputActivation() {...}

    Widget nilaMerahInputActivation() {...}

    Widget nilaHitamInputActivation() {...}

    Widget patinInputActivation() {...}

    Widget masInputActivation() {...}

    Widget submitButton() {...}

    Widget submitKeringButton() {...}
```

```

Widget previousSubmitButton() {...}

Widget previousNextButton() {...}

Widget nextButton() {...}

Widget previousButton() {...}

Widget deactivationInput() {...}

Widget destinationnNotActiveTransfer() {...}

Widget deactivationTransfer() {...}

return Scaffold(
    appBar: AppBar(
        backgroundColor: backgroundColor2,
        title: const Text("Entry Sortir"),
    ),
    backgroundColor: backgroundColor1,
    body: PageView(
        physics: const NeverScrollableScrollPhysics(),
        controller: pageController,
        children: [
            Obx(() {
                return ListView(
                    children: [
                        transferMethodInput(),
                        destinationPondInput(),
                        checkBoxFishTransfer(),
                        controller.isNilaHitamInput == true
                            ? nilaHitamInput()
                            : Container(),
                        controller.isNilaMerahInput == true
                            ? nilaMerahInput()
                            : Container(),
                        controller.isLeleInput == true ? leleInput() : Container(),
                        controller.isPatinInput == true ? patinInput() : Container(),
                        controller.isMasInput == true ? masInput() : Container(),
                        sampleLongInput(),
                        sampleWeightInput(),
                        controller.methodController.selected.value == "basah"
                            ? submitButton()
                            : nextButton(),
                        SizedBox(
                            height: 8,
                        )
                    ],
                );
            }),
            Obx(() {
                return ListView(
                    children: [

```

```

        controller.destinationIsActive == false
        ? destinationNotActiveTransfer()
        : deactivationInput(),
    controller.destinationIsActive == false
        ? checkBoxFish()
        : deactivationInput(),
    controller.isNilaHitamActivation == true
        ? nilaHitamInputActivation()
        : Container(),
    controller.isNilaMerahActivation == true
        ? nilaMerahInputActivation()
        : Container(),
    controller.isLeleActivation == true
        ? leleInputActivation()
        : Container(),
    controller.isPatinActivation == true
        ? patinInputActivation()
        : Container(),
    controller.isMasActivation == true
        ? masInputActivation()
        : Container(),
    controller.destinationIsActive == false
        ? waterHeightInput()
        : Container(),
    controller.destinationIsActive == false
        ? previousNextButton()
        : previousSubmitButton(),
    SizedBox(
        height: 8,
    )
],
);
}),
ListView(
    children: [
        deactivationTransfer(),
        deactivationInput(),
        previousSubmitButton(),
        SizedBox(
            height: 8,
        )
],
),
),
),
);
}
}

```

Code untuk halaman detail sortir

```

import 'package:fish/pages/treatment/treatment_detail_controller.dart';
import 'package:flutter/material.dart';

```

```

import 'package:fish/theme.dart';
import 'package:get/get.dart';
import '../controllers/fish_transfer/fish_transfer_detail_controller.dart';

class DetailSortirPage extends StatelessWidget {
  const DetailSortirPage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final SortirDetailController controller =
        Get.put(SortirDetailController());
    Widget transferDataRecap() {...}

    Widget detail() {...}

    Widget titleRecap() {...}

    Widget dataTreatment() {...}

    Widget detailSortir() {...}

    return Obx(() {
      if (controller.isLoading.value == false) {
        return Scaffold(
          appBar: AppBar(
            backgroundColor: backgroundColor2,
            title: const Text("Detail Sortir"),
          ),
          backgroundColor: backgroundColor1,
          body: ListView(
            children: [
              transferDataRecap(),
              detail(),
              titleRecap(),
              dataTreatment(),
              detailSortir(),
              SizedBox(
                height: 10,
              )
            ],
          ),
        );
      } else {
        return Center(
          child: CircularProgressIndicator(
            color: secondaryColor,
          ),
        );
      }
    });
  }
}

```

Membuat model class fitur sortir ikan

```

import 'package:fish/models/fish_harvested.dart';
import 'package:intl/intl.dart';
import 'package:fish/models/fish_model.dart';

class FishTransfer {
    String? id;
    String? origin_pond_id;
    String? origin_activation_id;
    String? destination_pond_id;
    String? destination_activation_id;
    String? transfer_method;
    String? transfer_type;
    num? sampleLong;
    num? sampleWeight;
    List<FishHarvest>? fishTransfer;
    String? transferAt;

    FishTransfer({
        required this.id,
        required this.origin_pond_id,
        required this.origin_activation_id,
        required this.destination_pond_id,
        required this.destination_activation_id,
        required this.transfer_type,
        required this.transfer_method,
        required this.sampleLong,
        required this.sampleWeight,
        required this.fishTransfer,
        this.transferAt,
    });

    factory FishTransfer.fromJson(Map<String, dynamic> json) {
        print(json);
        return FishTransfer(
            id: json['_id'],
            origin_pond_id: json['origin_pond_id'],
            origin_activation_id: json['origin_activation_id'],
            destination_pond_id: json['destination_pond_id'],
            destination_activation_id: json['destination_activation_id'],
            transfer_method: json['transfer_method'],
            transfer_type: json['transfer_type'],
            sampleLong: json['sample_long'],
            sampleWeight: json['sample_weight'],
            fishTransfer: FishHarvest.fromJsonList(json['fish']),
            transferAt: json['transfer_at'],
        );
    }

    String getGmtToNormalDate() {
        String stringDate = transferAt!;
        DateTime dateTime = DateFormat("yyyy-MM-dd hh:mm:ss").parse(stringDate);
    }
}

```

```

        String newStringDate = DateFormat("dd-MM-yyyy").format(dateTime);
        return newStringDate;
    }
}

```

Membuat network service untuk fitur sortir ikan

```

import 'dart:convert';
import 'dart:developer';
import 'package:fish/models/fish_transfer_model.dart';
import 'package:fish/service/url_api.dart';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;

class FishTransferService {
    Future<List<FishTransfer>> getFishTransferList() async {
        var url = Uri.parse(Urls.fishtransfer);
        var headers = {'Content-Type': 'application/json'};

        var response = await http.get(url, headers: headers);

        print(response.body);

        if (response.statusCode == 200) {
            var data = jsonDecode(response.body);
            List<FishTransfer> transferhistory = [];

            for (var item in data) {
                transferhistory.add(FishTransfer.fromJson(item));
            }
            // Treatment treatment = Treatment.fromJson(data[0]);
            // print(data[1]);

            print("ini leght transfer ${transferhistory.length}");
            return transferhistory;
        } else {
            throw Exception('Gagal Get Products!');
        }
    }

    Future<bool> postFishTransfer({
        required String? origin_pond_id,
        required String? destination_pond_id,
        required String? transfer_method,
        required String? transfer_type,
        required String? sample_weight,
        required String? sample_long,
        required List? fish,
    }) async {
        print({
            "origin_pond_id": origin_pond_id.toString(),
            "destination_pond_id": destination_pond_id.toString(),
        });
    }
}

```

```

    "transfer_method": transfer_method,
    "transfer_type": transfer_type,
    "sample_long": sample_long,
    "sample_weight": sample_weight,
    "fish": fish.toString()
  });
final response = await http.post(
  Uri.parse(Urls.fishtransfer),
  headers: {
    "Content-Type": "application/x-www-form-urlencoded",
  },
  encoding: Encoding.getByName('utf-8'),
  body: {
    "origin_pond_id": origin_pond_id.toString(),
    "destination_pond_id": destination_pond_id.toString(),
    "transfer_method": transfer_method,
    "transfer_type": transfer_type,
    "sample_long": sample_long,
    "sample_weight": sample_weight,
    "fish": fish.toString()
  },
);
if (response.statusCode == 200) {
  print(response.body);
  return true;
} else {
  print(response.body);
  return false;
}
}

Future<bool> postFishTransferKering(
  {required String? origin_pond_id,
  required String? destination_pond_id,
  required String? transfer_method,
  required String? transfer_type,
  required String? sample_weight,
  required String? sample_long,
  required List? fish,
  required List? fishstock,
  required List? fishharvested,
  required num? total_fish_harvested,
  required num? total_weight_harvested,
  String? water_level}) async {
print({
  "origin_pond_id": origin_pond_id.toString(),
  "destination_pond_id": destination_pond_id.toString(),
  "transfer_method": transfer_method,
  "transfer_type": transfer_type,
  "sample_long": sample_long,
  "sample_weight": sample_weight,
  "fish": fish.toString(),
});
}

```

```

    "fish_stock": fishstock.toString(),
    "fish_harvested": fishharvested.toString(),
    "total_weight_harvested": total_weight_harvested.toString(),
    "total_fish_harvested": total_fish_harvested.toString(),
    "water_level": water_level
  });
final response = await http.post(
  Uri.parse(Urls.fishtransfer),
  headers: {
    "Content-Type": "application/x-www-form-urlencoded",
  },
  encoding: Encoding.getByName('utf-8'),
  body: {
    "origin_pond_id": origin_pond_id.toString(),
    "destination_pond_id": destination_pond_id.toString(),
    "transfer_method": transfer_method,
    "transfer_type": transfer_type,
    "sample_long": sample_long,
    "sample_weight": sample_weight,
    "fish": fish.toString(),
    "fish_stock": fishstock.toString(),
    "fish_harvested": fishharvested.toString(),
    "total_weight_harvested": total_weight_harvested.toString(),
    "total_fish_harvested": total_fish_harvested.toString(),
    "water_level": water_level
  },
);
}

if (response.statusCode == 200) {
  print(response.body);
  return true;
} else {
  print(response.body);
  return false;
}
}

Future<bool> postTransfer(
  required String origin_pond_id,
  required String transfer_method,
  required String total_fish_harvested,
  required String total_weight_harvested,
  required String amountUndersize,
  required String amountOversize,
  required String amountNormal,
  required String sampleWeight,
  required String sampleLong,
  required String sampleAmount,
  required List<dynamic> transferList,
  required List<dynamic> fishDeath,
  required BuildContext ctx}) async {
List<dynamic> transferListPost = [];
for (var i in transferList) {

```

```

final fish = [];
for (var j in i["fish"]) {
    var k = json.decode(j);
    print("ini daata for ${k["type"]}");
    final datafish = {
        "type": k["type"],
        "amount": int.parse(k["amount"]),
        "weight": double.parse(k["weight"])
    };
    fish.add(datafish);
    print("ini fish baru $fish");
}
final datas = {
    "destination_pond_id": i["destination_pond_id"],
    "status": i["status"],
    "fish": fish,
    "sample_weight": double.parse(i["sample_weight"]),
    "sample_long": double.parse(i['sample_long']),
    "transfer_type": i["transfer_type"],
    if (i["status"] == "isNotActivated") ...{
        "water_level": int.parse(i["water_level"])
    }
};
transferListPost.add(datas);
}
print({
    "origin_pond_id": origin_pond_id.toString(),
    "fish_sort_type": transfer_method,
    "total_fish_harvested": total_fish_harvested,
    "total_weight_harvested": total_weight_harvested,
    "sample_long": sampleLong,
    "sample_amount": sampleAmount,
    "sample_weight": sampleWeight,
    "amount_oversize": amountOversize,
    "amount_undersized": amountUndersize,
    "amount_normal": amountNormal,
    "transfer_list": json.encode(transferListPost),
    "fish_death": json.encode(fishDeath)
});
final response = await http.post(
    Uri.parse(Urls.newfishtransfer),
    headers: {
        "Content-Type": "application/x-www-form-urlencoded",
    },
    encoding: Encoding.getByName('utf-8'),
    body: {
        "origin_pond_id": origin_pond_id.toString(),
        "fish_sort_type": transfer_method,
        "total_fish_harvested": total_fish_harvested,
        "total_weight_harvested": total_weight_harvested,
        "sample_long": sampleLong,
        "sample_amount": sampleAmount,
    }
);

```

```
        "sample_weight": sampleWeight,
        "amount_oversize": amountOversize,
        "amount_undersized": amountUndersize,
        "amount_normal": amountNormal,
        "transfer_list": json.encode(transferListPost),
        if (transfer_method == "kering") ...{
            "fish_death": json.encode(fishDeath)
        }
    },
);

if (response.statusCode == 200) {
    print(response.body);
    final snackBar = SnackBar(
        content: const Text('Sortir Ikan Berhasil!'),
        backgroundColor: Colors.green,
    );
    ScaffoldMessenger.of(ctx).showSnackBar(snackBar);
    return true;
} else {
    print(response.body);
    final snackBar = SnackBar(
        content: Text(response.body.toString()),
        backgroundColor: Colors.red,
    );
    ScaffoldMessenger.of(ctx).showSnackBar(snackBar);
}

print("gagal post");

return false;
}
}
}
```

L. Lampiran 12 Code Sprint 11 report

Code halaman login

```

import 'dart:convert';
import 'dart:developer';

import 'package:fish/controllers/authentication/login_controller.dart';
import 'package:fish/pages/authentication/register_page.dart';
// import 'package:rflutter_alert/rflutter_alert.dart';
import 'package:fish/pages/dashboard.dart';
import 'package:flutter/material.dart';
import 'package:fish/theme.dart';
import 'package:flutter/services.dart';
import 'package:get/get.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:fish/service/url_api.dart';
import 'package:shared_preferences/shared_preferences.dart';

import 'package:http/http.dart' as http;

import '../component/login_card_input.dart';

class LoginPage extends StatefulWidget {
    LoginPage({Key? key}) : super(key: key);
    @override
    State<LoginPage> createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
    late SharedPreferences prefs;
    final LoginController controller = Get.put(LoginController());
    @override
    void initState() {
        super.initState();
        initSharedPrefs();
    }

    void initSharedPrefs() async {
        prefs = await SharedPreferences.getInstance();
        inspect(prefs);
    }

    void login() async {
        final response = await http.post(
            Uri.parse(Urls.authentication),
            headers: {
                "Content-Type": "application/x-www-form-urlencoded",
            },
            encoding: Encoding.getByName('utf-8'),
            body: {
                "username": controller.usernameController.text,

```

```

        "password": controller.passwordController.text,
    },
);
var data = jsonDecode(response.body);
print(response.body);
if (response.statusCode == 200) {
    var myToken = data['access_token'];
    var identity = data['identity'];
    prefs.setString(
        'token',
        myToken,
    );
    prefs.setString('identity', identity.toString());
    // prefs.setString('identity', identity);
    Navigator.pushReplacement(
        context, MaterialPageRoute(builder: (context) => DashboardPage()));
    controller.usernameController.clear();
    controller.passwordController.clear();
} else {
    showDialog<String>(
        context: context,
        builder: (BuildContext context) => AlertDialog(
            title: const Text('Login Error',
                style: TextStyle(color: Colors.red)),
            content: const Text(
                'BreederID/Password salah',
                style: TextStyle(color: Colors.white),
            ),
            backgroundColor: backgroundColor1,
            shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.all(Radius.circular(16.0))),
            actions: <Widget>[
                TextButton(
                    onPressed: () => Navigator.pop(context, 'OK'),
                    child: const Text('OK'),
                ),
            ],
        )));
}
}

@Override
Widget build(BuildContext context) {

    Widget formInput() {...}

    Widget logo() {...}

    Widget footer() {...}

    Widget submitButton() {...}

    return Obx(() {

```

```

    if (controller.isLoading.value == false) {
      return Scaffold(
        backgroundColor: backgroundColor2,
        body: ListView(
          children: [
            SizedBox(
              height: 20,
            ),
            logo(),
            formInput(),
            SizedBox(
              height: 16,
            ),
            footer(),
            // submitButton(),
            SizedBox(
              height: 8,
            )
          ],
        ),
      );
    } else {
      return Center(
        child: CircularProgressIndicator(
          color: secondaryColor,
        ),
      );
    }
  });
}
}
}

```

Code untuk halaman register

```

import 'dart:convert';
import 'dart:developer';

import 'package:fish/controllers/authentication/register_controller.dart';
import 'package:fish/pages/dashboard.dart';
import 'package:flutter/material.dart';
import 'package:fish/theme.dart';
import 'package:flutter/services.dart';
import 'package:get/get.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:fish/service/url_api.dart';
import 'package:shared_preferences/shared_preferences.dart';

import 'package:http/http.dart' as http;

import '../component/login_card_input.dart';
import '../component/register_input.dart';
import '../component/register_input_next.dart';

```

```

import 'login_page.dart';

class RegisterPage extends StatefulWidget {
  RegisterPage({Key? key}) : super(key: key);
  @override
  State<RegisterPage> createState() => _RegisterPageState();
}

class _RegisterPageState extends State<RegisterPage> {
  late SharedPreferences prefs;
  final RegisterController controller = Get.put(RegisterController());

  final pageController = PageController(initialPage: 0);
  @override
  void dispose() {
    pageController.dispose();

    super.dispose();
  }

  void initState() {
    super.initState();
    initSharedPrefs();
    controller.getFarmData();
  }

  void initSharedPrefs() async {
    prefs = await SharedPreferences.getInstance();
    inspect(prefs);
  }

  void register() async {
    final response = await http.post(
      Uri.parseUrls.register,
      headers: {
        "Content-Type": "application/x-www-form-urlencoded",
      },
      encoding: Encoding.getByName('utf-8'),
      body: {
        "username": controller.usernameController.text,
        "password": controller.passwordController.text,
        "nik": controller.nikController.text,
        "name": controller.nameController.text,
        "phone": controller.phoneController.text,
        "hasFarm": controller.hasFarmController.selected.value,
        "farm_id": controller.farmIdSelected,
        "farm_name": controller.farmnameController.text,
        "breeder": controller.breedercountController.text,
        "address": controller.addressController.text,
        "coordinate": controller.coordinateController.text
      },
    );
    var data = jsonDecode(response.body);
  }
}

```

```

print(response.body);
if (response.statusCode == 200) {
  var myToken = data['access_token'];
  var identity = data['identity'];
  prefs.setString(
    'token',
    myToken,
  );
  prefs.setString('identity', identity.toString());
  Navigator.pushReplacement(
    context, MaterialPageRoute(builder: (context) => DashboardPage()));
  print(response.body);
  controller.usernameController.clear();
  controller.passwordController.clear();
  controller.addressController.clear();
  controller.breederCountController.clear();
  controller.coordinateController.clear();
  controller.phoneController.clear();
  controller.nameController.clear();
  controller.farmnameController.clear();
  controller.nikController.clear();
} else {
  showDialog<String>(
    context: context,
    builder: (BuildContext context) => AlertDialog(
      title: const Text('Register Error',
        style: TextStyle(color: Colors.red)),
      content: const Text(
        'BreederID Sudah Digunakan',
        style: TextStyle(color: Colors.white),
      ),
      backgroundColor: backgroundColor1,
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.all(Radius.circular(16.0))),
      actions: <Widget>[
        TextButton(
          onPressed: () => Navigator.pop(context, 'OK'),
          child: const Text('OK'),
        ),
      ],
    )));
  print(response.body);
}
}

@Override
Widget build(BuildContext context) {

Widget formInput() {...}

Widget form2Input() {...}

Widget logo() {...}

```

```

Widget footer() {...}

Widget submitButton() {...}

return Obx(() {
  if (controller.isLoading.value == false) {
    return Scaffold(
      backgroundColor: backgroundColor2,
      body: PageView(
        physics: const NeverScrollableScrollPhysics(),
        controller: pageController,
        children: [
          ListView(
            children: [
              SizedBox(
                height: 10,
              ),
              footer(),
              formInput(),
              SizedBox(
                height: 10,
              ),
            ],
          ),
          ListView(
            children: [
              SizedBox(
                height: 10,
              ),
              footer(),
              form2Input(),
              SizedBox(
                height: 10,
              ),
            ],
          ),
        ],
      );
  } else {
    return Center(
      child: CircularProgressIndicator(
        color: secondaryColor,
      ),
    );
  }
});
}

```

Code untuk halaman profile

```
import 'dart:convert';
import 'dart:developer';

import 'package:fish/controllers/authentication/register_controller.dart';
import 'package:fish/pages/dashboard.dart';
import 'package:flutter/material.dart';
import 'package:fish/theme.dart';
import 'package:get/get.dart';
import 'package:shared_preferences/shared_preferences.dart';
import '../controllers/authentication/profile_controller.dart';
import 'login_page.dart';

class ProfilePage extends StatelessWidget {
    ProfilePage({Key? key}) : super(key: key);
    @override
    final ProfileController controller = Get.put(ProfileController());

    @override
    Widget build(BuildContext context) {
        Widget logo() {...}

        Widget footer() {...}

        Widget profileList() {...}

        Widget logoutButton() {...}

        Widget submitButton() {...}

        return Obx(() {
            if (controller.isLoading.value == false) {
                return Scaffold(
                    backgroundColor: backgroundColor2,
                    body: ListView(
                        children: [
                            SizedBox(
                                height: 10,
                            ),
                            logo(),
                            SizedBox(
                                height: 16,
                            ),
                            profileList(),
                            logoutButton(),
                            SizedBox(
                                height: 8,
                            ),
                            // footer(),
                            // submitButton(),
                        ],
                    ),
                );
            }
        });
    }
}
```

```

    } else {
    return Center(
        child: CircularProgressIndicator(
            color: secondaryColor,
        ),
    );
}
));
}
}

```

Membuat model class user

```

import 'package:flutter/material.dart';
import 'package:intl/intl.dart';

class Breeder {
    String? id;
    String? farm_id;
    String? name;
    String? username;
    String? farm_name;
    String? nik;
    String? phone;
    String? address;

    Breeder({
        required this.id,
        required this.farm_id,
        required this.name,
        required this.username,
        required this.farm_name,
        required this.address,
        required this.nik,
        required this.phone,
    });

    factory Breeder.fromJson(Map<String, dynamic> json) {
        return Breeder(
            id: json['_id'],
            farm_id: json['farm_id'],
            farm_name: json['farm_name'],
            address: json['address'],
            name: json['name'],
            username: json['username'],
            nik: json['nik'],
            phone: json['phone'],
        );
    }

    static DateTime stringToDate(String dateString) {
        DateTime parseDate = DateFormat("dd-MM-yyyy").parse(dateString);
    }
}

```

```
    return parseDate;
}
}
```

Membuat model class farm

```
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';

class Farm {
    String? id;
    String? farm_name;
    String? breeder;
    String? address;
    String? coordinate;

    Farm({
        required this.id,
        required this.farm_name,
        required this.breeder,
        required this.address,
        required this.coordinate,
    });

    factory Farm.fromJson(Map<String, dynamic> json) {
        return Farm(
            id: json['_id'],
            farm_name: json['farm_name'],
            breeder: json['breeder'],
            address: json['address'],
            coordinate: json['coordinate']);
    }

    static DateTime stringToDate(String dateString) {
        DateTime parseDate = DateFormat("dd-MM-yyyy").parse(dateString);
        return parseDate;
    }
}
```

RIWAYAT HIDUP



GIAN CHIESA MAGHRIZA, Lahir di Bogor, 29 September 1999. Anak kedua dari pasangan Bapak Deden Setiaji dan Ibu Midayanti. Saat ini penulis tinggal di Perumahan Puspasari Elok Blok B4 no.12 RT03/10, Desa Puspasari, Kecamatan Citeureup, Kabupaten Bogor, Provinsi Jawa Barat.

Riwayat Hidup: Penulis mengawali pendidikan di TK Merpati Pos pada tahun 2005-2006. Kemudian melanjutkan pendidikan di SDN Puspasari 01 pada tahun 2006-2011. Selanjutnya penulis melanjutkan ke SMPN 1 Citeureup pada tahun 2011-2014. Kemudian melanjutkan pendidikan di SMAN 1 Cibinong pada tahun 2014-2018. Pada tahun 2018 penulis melanjutkan ke Universitas Negeri Jakarta (UNJ) di program studi Ilmu Komputer melalui jalur SBMPTN.

Riwayat Organisaasi: Selama di bangku perkuliahan, penulis merupakan anggota Kelompok Mahasiswa Peminat Fotografi UNJ. Penulis juga berpartisipasi dalam kegiatan pameran fotografi yang diadakan oleh KMPF UNJ. Penulis juga berpartisipasi dalam POMNAS XVI sebagai tim media koran dengan jobdesk sebagai fotografer tim media.

Riwayat Prestasi. Pada saat duduk di bangku SMA, penulis sering mengikuti kegiatan perlombaan Bahasa Jepang diikuti dengan perolehan juara III lomba cerdas cermat Bahasa Jepang di Jiyuu Matsuri UNJ. Selain itu, penulis juga pernah menjadi finalis Gemastik dalam kategori pengembangan software pada tahun 2022.