

**EKSPANSI APLIKASI AQUA BREEDING DENGAN
PENAMBAHAN FITUR INVENTARISASI UNTUK PENENTUAN
HARGA DASAR PRODUK PERIKANAN BERBASIS ANDROID**

Skripsi

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



*Mencerdaskan dan
Memartabatkan Bangsa*

**Akbar Maulana Alfatih
1313619003**

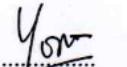
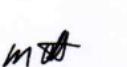
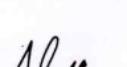
**PROGRAM STUDI ILMU KOMPUTER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS NEGERI JAKARTA**

2023

LEMBAR PERSETUJUAN HASIL SIDANG SKRIPSI

EKSPANSI APLIKASI AQUA BREEDING DENGAN PENAMBAHAN FITUR INVENTARISASI UNTUK PENENTUAN HARGA DASAR PRODUK PERIKANAN BERBASIS ANDROID

Nama : Akbar Maulana Alfatih
No. Registrasi : 1313619003

Penanggung Jawab	Nama	Tanda Tangan	Tanggal
Dekan	: Prof. Dr. Muktiningsih N, M.Si., NIP. 19640511 198903 2 001
Wakil Penanggung Jawab			
Wakil Dekan I	: Dr. Esmar Budi, S.Si., MT. NIP. 19720728 199903 1 002
Ketua	: Ir. Fariani Hermin Indiyah, MT. NIP. 19600211 198703 2 001		23-08-2023
Sekretaris	: Drs. Mulyono, M.Kom. NIP. 19660517 199403 1 003		23-08-2023
Penguji	: Dr. Ria Arafiah, M.Si. NIP. 19751121 200501 2 004		23-08-2023
Pembimbing I	: Muhammad Eka Suryana, M.Kom. NIP. 19851223 201212 1 002		22-08-2023
Pembimbing II	: Med Irzal, M.Kom. NIP. 19770615 200312 1 001		23-08-2023

Dinyatakan lulus ujian skripsi tanggal: 18 Agustus 2023

LEMBAR PERNYATAAN

Saya menyatakan dengan sungguh-sungguh bahwa skripsi dengan judul **“Ekspansi Aplikasi Aqua Breeding Dengan Penambahan Fitur Inventarisasi Untuk Penentuan Harga Dasar Produk Perikanan Berbasis Android”** yang telah saya susun sebagai syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Ilmu Komputer Universitas Negeri Jakarta adalah karya ilmiah saya dengan arahan dari dosen pembimbing.

Seluruh bahan dan data yang didapatkan dari penulis terdahulu yang sudah terpublikasikan yang tercantum dalam teks skripsi ini, telah tercantum di dalam Daftar Pustaka sesuai dengan etika, norma, dan kaidah penulisan ilmiah.

Apabila dikemudian hari diketemukan sebagian isi skripsi ini bukan hasil karya saya sendiri dalam beberapa bagian tertentu, saya bersedia mendapatkan sanksi pencabutan gelar akademik yang saya sanding dan sanksi-sanksi lainnya sesuai dengan peraturan perundang-undangan yang diberlakukan.

Jakarta, 02 Agustus 2023

Akbar Maulana Alfatih

HALAMAN PERSEMBAHAN

Untuk Keluargaku dan Diriku Sendiri.

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT, karena dengan rahmat dan karunia-Nya, penulis dapat menyelesaikan skripsi yang berjudul "**Ekspansi Aplikasi Aqua Breeding Dengan Penambahan Fitur Inventarisasi Untuk Penentuan Harga Dasar Produk Perikanan 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. Para petinggi di lingkungan FMIPA Universitas Negeri Jakarta.
2. Yth. Ibu Ria Arafiyah, M.Si selaku Koordinator Program Studi Ilmu Komputer.
3. Yth. Bapak Muhammad Eka Suryana, M.Kom selaku Dosen Pembimbing I yang telah membimbing, mengarahkan, serta memberikan saran dan koreksi terhadap skripsi ini.
4. Yth. Bapak Med Irzal, M.Kom selaku Dosen Pembimbing II yang telah membimbing, mengarahkan, serta memberikan saran dan koreksi terhadap skripsi ini.
5. Yth. Seluruh Dosen Ilmu Komputer Universitas Negeri Jakarta yang telah mendidik dan mengarahkan dari sisi akademik dalam penyusunan skripsi ini.
6. Orang tua penulis yang selama ini telah memberikan semangat, dukungan, serta doa kepada penulis dalam proses pembuatan skripsi ini.
7. Teman-teman Program Studi Ilmu Komputer 2019 yang telah mendukung dan menjadi penyemangat 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, 02 Agustus 2023

Akbar Maulana Alfatih

ABSTRAK

AKBAR MAULANA ALFATIH. Ekspansi Aplikasi Aqua Breeding Dengan Penambahan Fitur Inventarisasi Untuk Penentuan Harga Dasar Produk Perikanan Berbasis Android. Skripsi. Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Jakarta. 2023. Di bawah bimbingan Muhammad Eka Suryana, M.Kom dan Med Irzal, M.Kom.

Budidaya perikanan air tawar merupakan salah satu sumber perikanan yang ada di Indonesia. Dalam berbudidaya, tentunya penting untuk mencatat indikator-indikator inventaris budidaya ikan seperti pakan ikan, suplemen ikan, aset kolam, listrik pada kolam, serta benih ikan yang berguna untuk menentukan harga jual ikan. Penelitian ini bertujuan untuk memperluas aplikasi Aqua Breeding dengan menambahkan fitur inventarisasi yang dapat digunakan untuk mencatat penggunaan inventaris serta menentukan harga jual minimum ikan yang jujur. Data pada penelitian ini diambil dari hasil diskusi bersama pembudidaya ikan air tawar JFT (J Farm Technology) dan studi literatur dengan membaca jurnal-jurnal yang terkait dengan topik penelitian. Diskusi tersebut menghasilkan suatu user requirement yang menjadi pedoman dalam membuat web service pada backend serta penerapannya pada frontend mobile. Metode pengembangan sistem ini menggunakan metode Scrum dengan jumlah Sprint sebanyak lima Sprint serta teknologi yang digunakan adalah Flask dengan bahasa Python pada backend dan Flutter dengan bahasa Dart pada frontend. Hasil akhir dari penelitian ini adalah web service berupa REST API berserta dokumentasinya dan juga penerapannya pada aplikasi berbasis Android yang telah diuji dengan metode pengujian *unit testing* dan *User Acceptance Test* (UAT).

Kata kunci: *sistem inventarisasi, aplikasi mobile, transaksi ikan, budidaya perikanan modern, scrum*

ABSTRACT

AKBAR MAULANA ALFATIH. Aqua Breeding Application Expansion With Addition of Inventory Feature for Determining Base Price of Fishery Products Android Based. Thesis. Faculty of Mathematics and Natural Sciences, State University of Jakarta. 2023. Under the guidance of Muhammad Eka Suryana, M.Cs and Med Irzal, M.Cs.

Freshwater aquaculture is one of the sources of fisheries in Indonesia. In cultivating, of course it is important to record fish farming inventory indicators such as fish feed, fish supplements, pond assets, electricity in the pond, and fish seeds which are useful for determining the selling price of fish. This study aims to expand the Aqua Breeding application by adding an inventory feature that can be used to record inventory usage and determine an honest minimum selling price for fish. The data in this study were taken from discussions with freshwater fish cultivators JFT (J Farm Technology) and literature studies by reading journals related to the research topic. The discussion resulted in a user requirement that became a guide in creating web services on the backend and its application on the mobile frontend. The system development method uses the Scrum method with a total of five Sprints and the technology used is Flask with Python on the backend and Flutter with Dart on the frontend. The end result of this research is a web service in the form of a REST API along with its documentation and also its application to Android-based applications that have been tested using unit testing and User Acceptance Test (UAT) methods.

Kata kunci: *inventory system, mobile application, fish transaction, modern aquaculture, scrum*

DAFTAR ISI

LEMBAR PERSETUJUAN HASIL SIDANG SKRIPSI	ii
LEMBAR PERNYATAAN	iii
HALAMAN PERSEMBAHAN	iv
KATA PENGANTAR	vi
ABSTRAK	vii
ABSTRACT	viii
DAFTAR ISI	x
DAFTAR GAMBAR	xiii
DAFTAR TABEL	xiv
I PENDAHULUAN	1
A. Latar Belakang Masalah	1
B. Rumusan Masalah	4
C. Pembatasan Masalah	4
D. Tujuan Penelitian	5
E. Manfaat Penelitian	5
II KAJIAN PUSTAKA	6
A. Pengertian Persediaan dan Manajemen Persediaan	6
B. Jenis-jenis Manajemen Persediaan	7
C. Biaya Persediaan	8
D. Pengendalian Persediaan	8
E. Penentuan Harga Transfer	10

1.	Syarat Terpenuhinya Harga Transfer	10
2.	Tujuan Penentuan Harga Transfer	11
3.	Kebijakan Penentuan Harga Transfer	11
4.	Prinsip Dasar Penentuan Harga Transfer	12
F.	Aqua Breeding	15
G.	<i>Frontend dan Backend</i>	15
H.	Flutter	16
I.	Flask	21
J.	MongoDB	22
K.	REST API	22
L.	Scrum	23
M.	<i>Unit Testing</i>	24
N.	<i>User Acceptance Test (UAT)</i>	24
III METODOLOGI PENELITIAN		25
A.	Inventaris Budidaya Perikanan	25
B.	Metode Penentuan Harga Dasar	27
C.	Tahapan Penelitian	30
D.	Analisa Kebutuhan	30
E.	Perancangan Sistem	32
F.	Pengujian	35
IV HASIL DAN PEMBAHASAN		39
A.	Perancangan Sistem Dengan Scrum	39
1.	Sprint 1	39
2.	Sprint 2	48
3.	Sprint 3	56
4.	Sprint 4	79
5.	Sprint 5	138
B.	Kesimpulan Sprint	171

C. Pengujian Sistem	175
1. Unit Testing	176
2. User Acceptance Test	177
3. Kesimpulan Pengujian	178
V KESIMPULAN DAN SARAN	179
A. Kesimpulan	179
B. Saran	179
DAFTAR PUSTAKA	181
A LAMPIRAN	182
A. Transkrip Percakapan	182
B DAFTAR RIWAYAT HIDUP	183

DAFTAR GAMBAR

Gambar 2.1	Logo Aqua Breeding	15
Gambar 2.2	Halaman dengan Scaffold	18
Gambar 2.3	Halaman tanpa Scaffold	18
Gambar 2.4	List View	19
Gambar 2.5	Elevated Button	21
Gambar 2.6	Skema REST API	22
Gambar 3.1	Contoh Tabel Data Inventaris Pakan	25
Gambar 3.2	Contoh Tabel Data Inventaris Suplemen	26
Gambar 3.3	Contoh Tabel Data Inventaris Listrik	26
Gambar 3.4	Contoh Tabel Data Inventaris Benih	27
Gambar 3.5	Contoh Tabel Data Inventaris Aset	27
Gambar 3.6	Diskusi dengan Dinas Pertanian dan Perikanan Bogor	28
Gambar 3.7	Diskusi dengan Dinas Pertanian dan Perikanan Bogor	28
Gambar 3.8	Alur Tahapan Penelitian	30
Gambar 3.9	<i>Use Case</i> Aplikasi	32
Gambar 3.10	Tahapan Perancangan Sistem dengan Metode Scrum	33
Gambar 4.1	Skema Database Fitur Inventaris	40
Gambar 4.2	Integrasi Database Inventaris dengan Skema Database Iterasi 1	42
Gambar 4.3	Halaman Dashboard	43
Gambar 4.4	Halaman Menu Inventaris	43
Gambar 4.5	Halaman Data Inventaris Pakan	44
Gambar 4.6	Halaman Input Inventaris Pakan	44
Gambar 4.7	Halaman Detail Inventaris Pakan	44
Gambar 4.8	Halaman Data Inventaris Bahan Budidaya	45
Gambar 4.9	Halaman Input Inventaris Bahan Budidaya	45
Gambar 4.10	Halaman Detail Inventaris Bahan Budidaya	45

Gambar 4.11 Halaman Data Inventaris Tagihan Listrik	46
Gambar 4.12 Halaman Input Inventaris Tagihan Listrik	46
Gambar 4.13 Halaman Detail Inventaris Tagihan Listrik	46
Gambar 4.14 Halaman Data Inventaris Benih	47
Gambar 4.15 Halaman Input Inventaris Benih	47
Gambar 4.16 Halaman Detail Inventaris Benih	47
Gambar 4.17 Halaman Data Inventaris Aset	48
Gambar 4.18 Halaman Input Inventaris Aset	48
Gambar 4.19 Alur Inventaris Pakan	50
Gambar 4.20 Alur Inventaris Bahan Budidaya	51
Gambar 4.21 Alur Inventaris Listrik	52
Gambar 4.22 Alur Inventaris Benih	53
Gambar 4.23 Alur Inventaris Aset	54
Gambar 4.24 Update Skema Database Inventaris	55
Gambar 4.25 Sample Route Benih	57
Gambar 4.26 Package HTTP untuk Flutter	63
Gambar 4.27 Penambahan package HTTP pada Flutter	64
Gambar 4.28 Halaman Inventaris Benih	71
Gambar 4.29 Halaman Input Inventaris Benih	71
Gambar 4.30 Halaman Detail Inventaris Benih	71
Gambar 4.31 Halaman Aktivasi Kolam	72
Gambar 4.32 Halaman Aktivasi Kolam	72
Gambar 4.33 Sample Route Riwayat Benih	73
Gambar 4.34 Halaman Penggunaan Benih	79
Gambar 4.35 Sample Route Inventaris Pakan	81
Gambar 4.36 Halaman Inventaris Pakan	91
Gambar 4.37 Halaman Input Inventaris Pakan	91
Gambar 4.38 Halaman Detail Inventaris Pakan	91
Gambar 4.39 Sample Route Inventaris Suplemen	92

Gambar 4.40 Halaman Inventaris Suplemen	102
Gambar 4.41 Halaman Input Inventaris Suplemen	102
Gambar 4.42 Halaman Detail Inventaris Suplemen	102
Gambar 4.43 Sample Route Inventaris Listrik	103
Gambar 4.44 Halaman Inventaris Listrik	111
Gambar 4.45 Halaman Input Inventaris Listrik	111
Gambar 4.46 Halaman Detail Inventaris Listrik	111
Gambar 4.47 Sample Route Inventaris Aset	112
Gambar 4.48 Halaman Inventaris Aset	120
Gambar 4.49 Halaman Input Inventaris Aset	120
Gambar 4.50 Halaman Detail Inventaris Aset	120
Gambar 4.51 Sample Route Riwayat Pemakaian Pakan	121
Gambar 4.52 Halaman Riwayat Pemakaian Pakan	127
Gambar 4.53 Sample Route Riwayat Pemakaian Suplemen	128
Gambar 4.54 Halaman Riwayat Pemakaian Suplemen	134
Gambar 4.55 Halaman Entry Pakan	137
Gambar 4.56 Sample Route Merk Pakan	139
Gambar 4.57 Halaman Menu Inventaris Pakan	148
Gambar 4.58 Halaman Merk Pakan	148
Gambar 4.59 Halaman Detail Merk Pakan	148
Gambar 4.60 Halaman Treatment Kolam	153
Gambar 4.61 Halaman Treatment Kolam	153
Gambar 4.62 Halaman Panen	163
Gambar 4.63 Sample Route Pembukuan	165
Gambar 4.64 Halaman Dashboard	170
Gambar 4.65 Halaman Pembukuan	170
Gambar 4.66 Rapat dengan Pembudidaya Ikan	175
Gambar 2.1 Foto Penulis	183

DAFTAR TABEL

Tabel 3.1	Product Backlog	34
Tabel 3.2	Skenario Unit Testing	36
Tabel 3.3	Format <i>User Acceptance Test</i>	38
Tabel 4.1	Sprint 1 Backlog	39
Tabel 4.2	Sprint 2 Backlog	49
Tabel 4.3	Sprint 3 Backlog	57
Tabel 4.4	Sprint 4 Backlog	80
Tabel 4.5	Sprint 5 Backlog	139
Tabel 4.6	Sprint 1 Backlog	171
Tabel 4.7	Sprint 2 Backlog	172
Tabel 4.8	Sprint 3 Backlog	172
Tabel 4.9	Sprint 4 Backlog	173
Tabel 4.10	Sprint 5 Backlog	174
Tabel 4.11	Unit testing fitur inventarisasi.	176
Tabel 4.12	Unit testing integrasi inventarisasi dengan sistem.	176
Tabel 4.13	Format <i>User Acceptance Test</i>	177

BAB I

PENDAHULUAN

A. Latar Belakang Masalah

Perikanan merupakan suatu sumber penghasilan terbesar yang ada di Indonesia dikarenakan Indonesia sendiri disebut sebagai negara maritim yang memiliki arti negara kepulauan. Oleh karena itu, banyak penduduk di Indonesia yang bermata pencaharian sebagai pembudidaya ikan. Namun, jika terlalu banyak menangkap ikan akan menyebabkan *over fishing* yang membuat kemampuan bereproduksi ikan akan jauh lebih kecil daripada jumlah ikan hasil tangkapan. Hal ini akan menyebabkan langkanya spesies ikan tersebut dan berkurangnya angka produksi ikan. Dengan demikian, untuk mengatasi hal tersebut diperlukan budidaya perikanan yang berguna untuk menjaga ikan sampai masa panen tiba, serta dapat meningkatkan nilai ekonomi para pembudidaya ikan.

Dalam menjalankan budidaya perikanan, kebanyakan pembudidaya ikan masih melakukan cara manual dalam mengelola budidayanya. Hal ini tentunya kurang efektif dalam jangka panjang dan akan menyulitkan dalam pengelolaan budidayanya. Oleh karena itu, dalam penelitian yang dibuat oleh (Lin, 2019) dan (Ouyang, 2021) dapat berguna dalam menerapkan budidaya perikanan modern.

Yi-Bing Lin dan timnya membuat *smart aquarium* yang bertujuan untuk meningkatkan kualitas akuarium yang bernama FishTalk. FishTalk memungkinkan sebuah sensor pada akuarium untuk menggerakan aktuator secara real time. Kegunaan dari *smart aquarium* ini seperti sistem pemberian pakan otomatis dan pengendalian air dalam kolam secara otomatis. (Lin, 2019)

Sementara itu, Bing Ouyang dan timnya membuat sebuah sistem yang dibentuk dan digunakan untuk monitoring serta *decision making* pada tambak perikanan, sistem ini dinamakan HAUCS (*Hybrid Aerial Underwater Robotic System*). Pemantauan ini dilakukan dengan memanfaatkan sistem robotik, mesin,

dan operator manusia. Tujuan dibentuknya HAUCS ini adalah untuk meringankan pekerjaan manusia dari tugas yang berat, terlalu banyak biaya, dan memakan waktu dalam operasi pelaksanaan budidaya *aquaculture* melalui platform pemanfaatan sistem robotik. (Ouyang, 2021)

Dari kedua penelitian diatas, dapat disimpulkan bahwa alat yang digunakan dapat bermanfaat bagi para pembudidaya ikan karena dapat mempermudah pengelolaan budidaya. Namun, tentunya alat dan bahan yang dibutuhkan cukup banyak dan pasti mematok harga yang tidak sedikit.

Oleh karena itu, penelitian yang dilakukan oleh (Chen, 2020) dan timnya mungkin dapat mengatasi masalah tersebut. Penelitian ini bertujuan untuk membuat big data dengan *framework* SpringBoot dan Java Persistence API (JPA) yang didalamnya terdapat data kualitas air pada setiap perkembangbiakan ikan ternak. Platform ini dapat digunakan untuk memprediksi kualitas air dari setiap kolam dan memberikan notifikasi langsung ketika ada masalah pada kolam tersebut. Namun, penelitian ini hanya berfokus pada pendataan kualitas air saja sehingga rincian lain dari budidaya tersebut masih belum lengkap. (Chen, 2020)

Tapi, tidak seperti dua penelitian yang sudah dirujuk sebelumnya, penelitian (Chen, 2020) ini berbasis aplikasi sehingga tidak ada biaya peralatan tambahan. Dengan demikian, pembudidaya ikan akan lebih terbantu jika terdapat aplikasi yang dapat membantu mereka dalam mengembangkan budidayanya tanpa perlu mengeluarkan biaya tambahan.

Pada penelitian yang dilakukan oleh (Hadi, 2021), (Maghriza, 2022) dan (Rahmanto, 2022), mereka membuat suatu aplikasi bernama *Aqua Breeding* yang berfungsi untuk mencatat pendetailan dari setiap budidaya para pembudidaya ikan. Detail yang dimaksud seperti pencatatan pakan ikan, pencatatan angka kematian ikan, pengendalian kualitas air, dan pencatatan lainnya yang berhubungan pada musim budidaya ikan tersebut. Aplikasi ini tentunya dapat membantu para pembudidaya ikan dan juga dapat meningkatkan ekonomi pembudidaya ikan sejalan dengan lancarnya musim budidaya.

Penelitian yang terkait dalam aplikasi tersebut adalah penelitian Fadhil Perwira Hadi yang berjudul “Rancang Bangun Web Service dan Website sebagai Storage Engine dan Monitoring Data Sensing untuk Budidaya Ikan Air Tawar” menghasilkan suatu sistem web service yang dapat menerima data yang dikirimkan oleh *embedded device*, dengan menerapkan konsep IoT (Hadi, 2021). *Web service* tersebut kemudian dilanjutkan dengan penelitian Andri Rahmanto dengan judul “Perancangan Arsitektur Aplikasi Budidaya Perikanan Modern pada Backend yang bertanggung jawab dalam melayani Transaksi Query Webservice dengan menggunakan Teknologi Flask Microservice”. *Web service* ini menghasilkan *output* berupa arsitektur aplikasi budidaya perikanan modern pada *backend* berupa *endpoint* yang dapat digunakan untuk pendataan budidaya perikanan air tawar (Rahmanto, 2022). Dalam pengolahan *backend* ini, Gian Chiesa Maghriza dengan penelitiannya yang berjudul “Perancangan Frontend Aplikasi Pendukung Teknologi Perikanan Modern dengan menggunakan Framework Flutter yang mentarget Multi Platform” membuat *user interface* serta konfigurasi fitur pencatatan dari aplikasi teknologi perikanan modern. Fitur-fitur yang ada pada aplikasi ini didasari pada penggunaan *endpoint* yang sudah disediakan pada *backend* buatan Andri (Maghriza, 2022). Namun pada aplikasi tersebut masih terdapat kekurangan seperti belum tersedia fitur inventarisasi sebagai *storage* dalam berbudidaya.

Beberapa masalah yang dialami oleh pembudidaya ikan antara lain mayoritas distributor memainkan timbangan, ketimpangan harga penawaran oleh distributor pada petani produsen, dan tidak ada data jelas terkait jumlah demand dan besarnya supply. Untuk mengatasi permasalahan tersebut, dibuatlah sistem yang melakukan tracking aktifitas Budidaya untuk mengetahui data supply di tingkat pembudidaya. Sistem ini sudah dibuat oleh (Rahmanto, 2022) dan (Maghriza, 2022). Lalu, perlu ada nya sistem yang melakukan tracking pengeluaran yang dikeluarkan pembudidaya untuk mengetahui harga dasar nilai jual ikan. Sistem ini akan dibuat pada penelitian ini yaiut berupa sistem inventaris. Sistem inventaris ini berguna untuk mencatat dan melihat penggunaan supply selama masa berbudidaya sehingga

para pembudidaya ikan dapat melihat dengan jelas data terkait besarnya supply. Dari fitur ini, para pembudidaya ikan juga dapat menentukan harga dasar penjualan ikan berdasarkan pengeluaran yang dilakukan selama masa budidaya. Oleh karena itu, pembudidaya ikan dapat menjual ikannya dengan harga yang tepat dan adil sebab sering terjadinya ketimpangan harga pada saat transaksi. Kemudian, diperlukan sistem interkoneksi pembudidaya yang menjadi titik pertemuan supply dan demand tersebut. Sistem ini akan dilanjutkan pada penelitian selanjutnya.

Oleh karena itu, penelitian ini bertujuan untuk melengkapi sistem untuk memecahkan masalah-masalah tersebut dengan menambahkan sistem inventarisasi pada aplikasi *Aqua Breeding* ini untuk menentukan harga dasar pada produk perikanan. Dengan demikian, harga dasar tersebut dapat digunakan oleh pembudidaya ikan untuk penjualan hasil panen mereka. Selain itu, fitur inventarisasi ini juga dapat membantu para pembudidaya ikan dalam mengolah dan mengontrol kebutuhan serta pengeluaran dalam setiap musim budidaya. Berdasarkan fitur baru yang sudah dijelaskan sebelumnya, aplikasi ini diharapkan dapat membantu para pembudidaya ikan untuk berbudidaya dalam hal penentuan harga dasar dan pengendalian kebutuhan saat budidaya berlangsung.

B. Rumusan Masalah

Dari uraian latar belakang di atas, perumusan masalah pada penelitian ini ialah “Bagaimana rancangan sistem inventaris yang digunakan untuk menentukan harga dasar dan harga jual minimum dari produk perikanan berbasis Android? ”.

C. Pembatasan Masalah

Pembatasan masalah pada penelitian ini antara lain:

1. Aplikasi hanya menentukan harga jual minimum dari produk perikanan.
2. Formula penentuan harga dasar ikan masih berupa hipotesis.

D. Tujuan Penelitian

Penelitian ini dilakukan dengan tujuan untuk membuat aplikasi perikanan berbasis Android dengan sistem inventaris untuk menentukan harga jual minimum dari produk perikanan.

E. Manfaat Penelitian

1. Bagi penulis

Meningkatkan pengetahuan sistem inventaris pada produk perikanan, menambah pengalaman dalam mengembangkan aplikasi, memperoleh gelar sarjana di bidang Ilmu Komputer, serta menjadi media untuk penulis dalam mengaplikasikan ilmu yang didapat dari kampus.

2. Bagi Universitas Negeri Jakarta

Menjadi pedoman untuk penelitian di masa depan, dan dapat memberikan panduan bagi mahasiswa program studi Ilmu Komputer tentang penentuan harga dasar pada produk perikanan dengan sistem inventaris.

3. Bagi masyarakat

Membantu masyarakat yang ingin dan sedang menggeluti bidang budidaya perikanan dalam proses penentuan harga dasar pada produk perikanan.

BAB II

KAJIAN PUSTAKA

A. Pengertian Persediaan dan Manajemen Persediaan

Pada buku Dasar-Dasar Manajemen (Sim, 2022), dijelaskan bahwa persediaan adalah sebuah stok aset yang dimiliki oleh perusahaan. Aset ini dapat berupa bahan mentah, bahan baku, barang jadi, barang dalam proses, hingga bahan pembantu. Persediaan merupakan aset yang berharga, karena berkaitan dengan proses produksi. Persediaan yang tidak teratur dapat menyebabkan kerugian, sehingga menerapkan manajemen persediaan dalam suatu bisnis merupakan hal yang penting.

Manajemen persediaan merupakan suatu cara untuk melakukan pengawasan, kontrol, pengelolaan terhadap persediaan yang dimiliki oleh perusahaan. Berbagai macam kegiatan yang berkaitan dengan memperoleh, menyimpan, hingga menggunakan persediaan merupakan bagian dari manajemen persediaan.

Manajemen persediaan memiliki beberapa fungsi, yaitu:

1. Mencegah terjadinya kekurangan persediaan.
2. Mencegah barang dari supplier tidak sesuai kebutuhan.
3. Memastikan proses produksi berjalan dengan lancar.
4. Mengantisipasi permintaan yang mendadak.
5. Menyesuaikan pembelian dengan jadwal produksi.

Selain beberapa fungsi yang sudah disebutkan diatas, Manajemen persediaan juga memiliki tujuan. Beberapa tujuan dari manajemen persediaan adalah sebagai berikut.

1. Mengantisipasi kenaikan harga bahan baku.

2. Memastikan persediaan selalu tersedia.
3. Mengurangi resiko bahan baku yang datang terlambat.
4. Menjaga jumlah persediaan tetap stabil.
5. Mengantisipasi kemungkinan adanya perubahan dari segi penawaran ataupun permintaan.

B. Jenis-jenis Manajemen Persediaan

Manajemen persediaan dibagi menjadi beberapa jenis, diantaranya:

1. Bahan Mentah

Bahan mentah atau bahan baku merupakan bahan utama atau dasar dari dibuatnya suatu produk. Tanpa adanya bahan baku, maka produk tidak bisa masuk ke tahap produksi. Oleh karena itu manajemen persediaan diperlakukan untuk mengelola bahan baku agar selalu tersedia dan siap untuk diproses.

2. Barang Setengah Jadi

Barang setengah jadi atau barang dalam proses merupakan barang yang belum sepenuhnya bisa digunakan, sehingga perlu untuk diproses lebih lanjut untuk menjadi barang jadi, yang nantinya siap untuk digunakan. Dalam hal ini, manajemen persediaan digunakan untuk menghitung banyaknya barang setengah jadi tersebut untuk memenuhi kebutuhan pasar.

3. Barang Jadi

Barang jadi merupakan barang yang sudah siap untuk dijual. Manajemen persediaan berguna untuk mengatur pengiriman barang tersebut ke pasar sehingga keadaan produk di pasar tetap stabil.

C. Biaya Persediaan

Penetapan biaya persediaan atau evaluasi persediaan memungkinkan perusahaan untuk memberikan nilai moneter untuk barang-barang dalam persediaan mereka. Inventaris perusahaan seringkali merupakan aset terbesarnya dan pengukuran yang tepat untuk memastikan keakuratan laporan keuangan.

Untuk menentukan biaya persediaan, diperlukan lima langkah-langkah sebagai berikut.

1. Menentukan periode waktu tertentu yang dimana perlu menemukan nilai inventaris.
2. Memastikan persediaan selalu tersedia.
3. Mengurangi resiko bahan baku yang datang terlambat.
4. Menjaga jumlah persediaan tetap stabil.
5. Mengantisipasi kemungkinan adanya perubahan dari segi penawaran ataupun permintaan.

D. Pengendalian Persediaan

Pengendalian persediaan adalah suatu tahapan yang digunakan untuk menyelesaikan masalah terkait dengan pengendalian barang pada perusahaan.

Persediaan yang terlalu berlebihan akan merugikan, karena akan lebih banyak modal yang diperlukan.

Menurut Sunyoto (2012:225), Sistem pengendalian persediaan merupakan serangkaian pengendalian untuk menentukan tingkat persediaan yang harus dijaga. Sistem ini menentukan dan menjamin tersedianya persediaan yang tepat dalam kualitas dan waktu yang tepat. Jika persediaan terlalu sedikit dapat mengakibatkan resiko terjadinya kekurangan persediaan atau bisa dibilang *stockout*. Bila persediaan dilebihkan, maka biaya penyimpanan dan modal yang diperlukan akan bertambah.

Sebaliknya, jika persediaan dikurangi maka akan mengalami *stockout* (kehabisan barang).

Menurut Assauri (2004), Pengendalian persediaan dapat dikatakan sebagai suatu kegiatan untuk menentukan tingkat dan komposisi dari persediaan sehingga perusahaan dapat melindungi kelancaran produksi dan penjualan serta kebutuhan-kebutuhan perusahaan dengan efisien.

Dengan kata lain, pengendalian persediaan akan mempermudah perusahaan untuk memproduksi barang, disimpan di gudang dan sampai ke konsumen. Persediaan yang terlalu besar (*overstock*) merupakan pemborosan karena menyebabkan tingginya beban biaya untuk inventaris barang-barang tersebut, sementara jika persediaan terlalu kecil maka dapat menyebabkan proses produksi terhenti sehingga konsumen akan pergi karena permintaannya tidak terpenuhi.

Agar pengendalian persediaan dapat dilakukan dengan maksimal, menurut Assauri (2004:176) ada faktor-faktor yang harus dipertimbangkan dalam menjalankan pengendalian persediaan, antara lain:

1. Adanya fasilitas pergudangan yang cukup luas dan teratur
2. Adanya sistem administrasi pencatatan dan pemeriksaan atas penerimaan dan pengeluaran barang
3. Sumber daya yang menguasai sistem administrasi pengendalian persediaan yang digunakan perusahaan
4. Perencanaan untuk mengganti barang yang telah digunakan dan barang yang sudah lama berada dalam gudang sehingga usang
5. Informasi dari bagian produksi tentang sifat teknis barang, daya tahan produk dan lamanya produksi, untuk melakukan perencanaan pengendalian persediaan
6. Informasi dari bagian penjualan tentang tingkat penjualan produk perusahaan, sehingga bagian persediaan bisa menentukan besarnya persediaan yang

seharusnya ada sehingga tidak terjadi kekurangan persediaan yang mengakibatkan pesanan konsumen tidak terpenuhi.

E. Penentuan Harga Transfer

Pada buku *Management Control* (Supitriyani, 2022), dijelaskan bahwa penentuan harga transfer atau *transfer pricing* merupakan proses harga penentuan harga yang ditetapkan dalam transaksi penjualan dan pembelian diantara berbagai unit organisasi pada kelompok perusahaan atau instansi yang sama.

Dalam menentukan harga transfer, prinsip dasarnya adalah bahwa harga transfer sebaiknya serupa dengan harga yang akan dikenakan seandainya produk tersebut dijual ke konsumen luar atau dibeli dari pemasok luar. Ketika suatu perusahaan membeli atau menjual produk

1. Syarat Terpenuhinya Harga Transfer

Menurut (Silalahi et al., 2019), syarat-syarat yang harus dijalankan agar terpenuhinya harga transfer adalah:

1. Sistem harus dapat memberikan informasi yang relevan yang dibutuhkan oleh suatu pusat laba untuk dapat menemukan trade-off yang optimum antara biaya dan pendapatan perusahaan.
2. Laba yang dihasilkan harus dapat menggambarkan dengan baik pengaturan trade-off antara biaya-pendapatan yang telah ditetapkan. Setiap pusat laba harus dapat memaksimalkan laba perusahaan dengan jalan memaksimalkan laba divisinya.
3. Tingkat laba yang diperlihatkan oleh masing-masing pusat laba harus dapat mencerminkan besarnya kontribusi laba dari masing-masing pusat laba terhadap laba perusahaan secara keseluruhan.

2. Tujuan Penentuan Harga Transfer

Menurut (Silalahi et al., 2019), harga transfer harus dirancang sedemikian rupa sehingga dapat mencapai tujuan berikut :

1. Memberikan informasi yang relevan kepada masing-masing unit usaha untuk menentukan imbal balik yang optimum antara biaya dan pendapatan perusahaan.
2. Menghasilkan keputusan yang selaras dengan cita-cita, maksudnya sistem harus dirancang sedemikian rupa sehingga keputusan yang meningkatkan laba unit usaha juga akan meningkatkan laba perusahaan.
3. Membantu pengukuran kinerja ekonomi dari unit usaha individual.
4. Sistem tersebut harus mudah dimengerti dan dikelola.

3. Kebijakan Penentuan Harga Transfer

Menurut (Hansen and Mowen, 2009), dalam penyusunan sebuah kebijakan penetapan harga transfer, kedua pandangan dari divisi penjual dan divisi pembeli harus dipertimbangkan. Pendekatan biaya peluang (opportunity cost approach) mencapai tujuan tersebut dengan mengidentifikasi harga minimum yang ingin diterima divisi penjual dan harga maksimum yang ingin dibayar divisi pembeli. Berikut harga-harga yang ditetapkan di setiap divisi:

1. Harga transfer minimum

Harga transfer minimum adalah harga transfer yang akan membuat keadaan divisi penjual tidak menjadi lebih buruk jika barang yang dijual pada divisi internal daripada dijual pada pihak luar.

2. Harga transfer maksimum

Harga transfer maksimum adalah harga transfer yang akan membuat keadaan divisi pembeli tidak menjadi lebih buruk jika suatu input dibeli dari divisi internal daripada jika barang yang sama dibeli secara eksternal.

4. Prinsip Dasar Penentuan Harga Transfer

Menurut (Anthony and Govindarajan, 2018), masalah penentuan harga transfer sebenarnya merupakan penentuan harga pada umumnya, dengan sedikit modifikasi untuk mempertimbangkan faktor-faktor tertentu yang unik dalam transaksi internal. Prinsip dasarnya adalah bahwa harga transfer sebaiknya serupa dengan harga yang akan dikenakan seandainya produk tersebut dijual ke konsumen luar atau dibeli dari pemasok luar.

Ketika suatu pusat laba di suatu perusahaan membeli produk dan menjual ke satu sama lain, maka dua keputusan yang harus diambil untuk setiap produk adalah:

1. Apakah perusahaan harus memproduksi sendiri produk tersebut atau membelinya dari pemasok luar. Hal ini merupakan keputusan sourcing.
2. Jika diproduksi secara internal, pada tingkat harga berapakah produk tersebut akan ditransfer antarpusat laba. Hal ini merupakan keputusan harga transfer.

Sistem harga transfer dapat bervariasi dari yang paling sederhana sampai yang paling rumit, tergantung dari sifat usahanya. Berikut merupakan beberapa jenis situasi dalam menentukan sistem harga transfer.

1. Situasi Ideal

Menurut (Anthony and Govindarajan, 2018), harga transfer berdasarkan harga pasar akan menghasilkan keselarasan jika kondisi-kondisi berikut ada, yaitu :

- (a) Orang-orang Kompeten.
- (b) Atmosfer yang baik.
- (c) Harga Pasar.

- (d) Kebebasan Memperoleh Sumber Daya.
- (e) Informasi Penuh.
- (f) Negoisasi

2. Hambatan dalam Perolehan Sumber Daya

Seorang manajer pembelian bebas mengambil keputusan sourcing. Demikian halnya dengan manajer penjualan, ia harus bebas untuk menjual produknya ke pasar yang paling menguntungkan. Menurut (Anthony and Govindarajan, 2018), akibat-akibat yang terjadi jika para manajer pusat laba tidak memiliki kebebasan dalam mengambil keputusan sourcing adalah sebagai berikut.

- (a) Pasar yang terbatas

Beberapa alasan pasar terbatas bagi pusat laba (pembeli dan penjual):

- i. Keberadaan kapasitas internal mungkin membatasi pengembangan penjualan eksternal.
- ii. Jika suatu perusahaan merupakan produsen tunggal dari produk yang terdifferensiasi, tidak ada sumber dari luar.
- iii. Jika perusahaan telah melakukan investasi yang besar, cenderung tidak akan menggunakan sumber daya dari luar kecuali harga jual di luar mendekati biaya variabel perusahaan.

Dalam kondisi pasar yang terbatas, harga transfer yang paling memenuhi persyaratan sistem pusat laba adalah harga kompetitif.

Perusahaan dapat mengetahui tingkat harga kompetitif jika perusahaan tersebut tidak membeli atau menjual produknya ke pasar bebas melalui cara-cara dibawah ini.

- i. Jika ada harga pasar diterbitkan, maka harga tersebut dapat digunakan untuk menentukan harga transfer.
- ii. Harga pasar mungkin ditentukan berdasarkan penawaran.

iii. Jika pusat laba pembelian membeli produk yang serupa dari pasar luar/bebas maka pusat laba tersebut dapat meniru untuk harga kompetitif untuk produk-produk eksklusifnya.

(b) Kelebihan atau Kekurangan Kapasitas Industri

Jika pusat laba penjualan tidak dapat menjual seluruh produk ke pasar bebas atau memiliki kapasitas produksi yang berlebih. Perusahaan mungkin tidak akan mengoptimalkan labanya jika pusat laba pembelian membeli produk dari pemasok luar sementara sementara kapasitas produksi di dalam masih memadai. Dan sebaliknya, jika pusat laba pembelian tidak dapat memperoleh produk yang diperlukan dari luar sementara pusat laba penjualan menjual produknya ke pihak luar. Situasi ini terjadi ketika terdapat kekurangan kapasitas produksi di dalam industri, sehingga pusat laba pembelian terhalang dan laba perusahaan tidak optimal.

Meskipun ada hambatan dalam perolehan sumber daya, harga pasar tetap merupakan harga transfer yang baik. Meskipun demikian, jika tidak ada cara untuk memperkirakan harga kompetitif, pilihan lainnya adalah mengembangkan harga transfer berdasarkan biaya (cost based transfer price). Biasanya, perusahaan akan mengeliminasi unsur iklan, pendanaan, atau pengeluaran lain yang tidak dikeluarkan oleh pihak penjual dalam transaksi internal saat penentuan harga transfer.

F. Aqua Breeding



Gambar 2.1: Logo Aqua Breeding

Aqua Breeding merupakan aplikasi yang digunakan pembudidaya ikan untuk mengontrol budidaya ikan mereka. Aplikasi ini memiliki fitur seperti mencatat pemberian pakan, perhitungan kualitas air, mencatat total panen, serta aktivitas budidaya ikan lainnya. Dari logo pada **Gambar 2.1**, dapat diartikan aplikasi Aqua Breeding memiliki arti menyatukan para pembudidaya ikan menjadi satu kesatuan dalam aplikasi ini.

G. *Frontend dan Backend*

Dalam pengembangan aplikasi, terdapat 2 sisi pengembangan yaitu *frontend* dan *backend*. *Frontend* merupakan bagian yang ditampilkan kepada user seperti contoh halaman dashboard, menu aplikasi, dan sebagainya. Oleh sebab itu, sisi *frontend* ini juga bisa dibilang sebagai *user-side* atau *client-side*.

Sementara itu, *backend* merupakan bagian yang mengurus koneksi antara server dengan *database* aplikasi. *Backend* juga bertugas untuk membuat penghubung antara aplikasi dengan server dalam bentuk *endpoint* yang nantinya akan digunakan pada sisi *frontend* untuk ditampilkan pada aplikasi. Oleh karena itu, sisi *backend* bisa dibilang sebagai *server-side*.

H. Flutter

Flutter merupakan salah satu *framework* yang digunakan dalam pengembangan aplikasi *mobile* (Flutter, 2017). Flutter dikembangkan dengan bahasa pemrograman Dart yang dirancang oleh Lars Bak dan Kasper Lund dan memiliki struktur berbasis *class* dan berorientasi terhadap objek. *Framework* ini diresmikan pada tahun 2015 oleh perkumpulan Dart *developer summit* dan merilis versi stabilnya yaitu Flutter 1.0 pada tahun 2018 di acara Flutter Live. Pengembangan *framework* ini tergolong cukup besar karena bersifat *open source* sehingga banyak komunitas dan orang-orang yang ikut mengembangkan *framework* ini.

Beberapa keunggulan penggunaan Flutter sebagai *framework* pengembangan aplikasi *mobile* adalah sebagai berikut.

1. *Multiplatform*

Multiplatform berarti bahwa *framework* ini dapat digunakan untuk mengembangkan dua sisi mobile yaitu Android dan iOS dengan satu basis kode. Hal ini tentunya dapat mempersingkat waktu serta mengurangi biaya pada bagian *development*.

2. *Open Source*

Open Source berarti bahwa *framework* ini dapat dimodifikasi oleh pengguna sehingga user juga dapat berkontribusi dalam pengembangan *framework* ini.

3. Dokumentasi lengkap

Dokumentasi lengkap *framework* Flutter dapat diakses pada situs resmi Flutter di <https://flutter.dev/>.

Dalam Flutter, bagian-bagian yang membentuk sebuah halaman disebut sebagai widget. Beberapa widget Flutter yang umum digunakan dalam pembuatan aplikasi adalah sebagai berikut.

1. Scaffold

Scaffold merupakan bagian utama dari sebuah halaman pada Flutter. Bisa dibilang Scaffold ini adalah *parent* dari semua widget. Berikut contoh penerapan Scaffold pada Flutter.

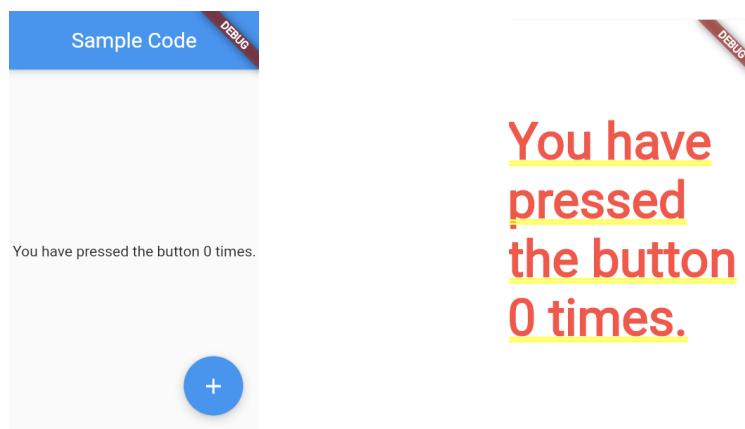
```
1 import 'package:flutter/material.dart';
2
3 /// Flutter code sample for [Scaffold].
4
5 void main() => runApp(const ScaffoldExampleApp());
6
7 class ScaffoldExampleApp extends StatelessWidget {
8   const ScaffoldExampleApp({super.key});
9
10  @override
11  Widget build(BuildContext context) {
12    return const MaterialApp(
13      home: ScaffoldExample(),
14    );
15  }
16}
17
18 class ScaffoldExample extends StatefulWidget {
19   const ScaffoldExample({super.key});
20
21   @override
22   State<ScaffoldExample> createState() => _ScaffoldExampleState();
23 }
24
25 class _ScaffoldExampleState extends State<ScaffoldExample> {
26   int _count = 0;
27
28   @override
29   Widget build(BuildContext context) {
30     return Scaffold(
31       appBar: AppBar(
32         title: const Text('Sample Code'),
33       ),
34       body: Center(child: Text('You have pressed the button $_count times.')),
35       floatingActionButton: FloatingActionButton(
36         onPressed: () => setState(() => _count++),
37         tooltip: 'Increment Counter',
38         child: const Icon(Icons.add),
```

```

39     ) ,
40     );
41   }
42 }
43
44

```

Berikut merupakan hasil layout dari code sebelumnya.



Gambar 2.2:
Halaman dengan Scaffold

Gambar 2.3:
Halaman tanpa Scaffold

Dapat dilihat pada perbandingan **Gambar 2.2** dan **Gambar 2.3**, tanpa menggunakan Scaffold maka halaman tidak akan terproses dengan baik.

2. List View

List View merupakan salah satu widget yang digunakan untuk menampilkan list dari data. Berikut contoh penggunaan List View.

```

1  ListView(
2    padding: const EdgeInsets.all(8),
3    children: <Widget>[
4      Container(
5        height: 50,
6        color: Colors.amber[600],
7        child: const Center(child: Text('Entry A')),
8      ),
9      Container(

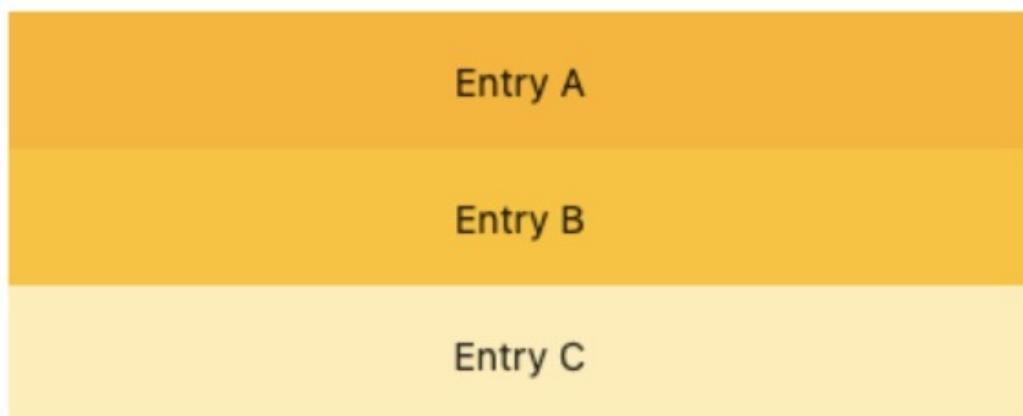
```

```

10    height: 50,
11    color: Colors.amber[500],
12    child: const Center(child: Text('Entry A')),
13  ),
14  Container(
15    height: 50,
16    color: Colors.amber[100],
17    child: const Center(child: Text('Entry B')),
18  ),
19 ],
20 )
21

```

Berikut merupakan hasil layout dari code sebelumnya.



Gambar 2.4: List View

3. Elevated Button

Elevated Button merupakan salah satu widget button yang digunakan pada Flutter. Berikut contoh penggunaan Elevated Button.

```

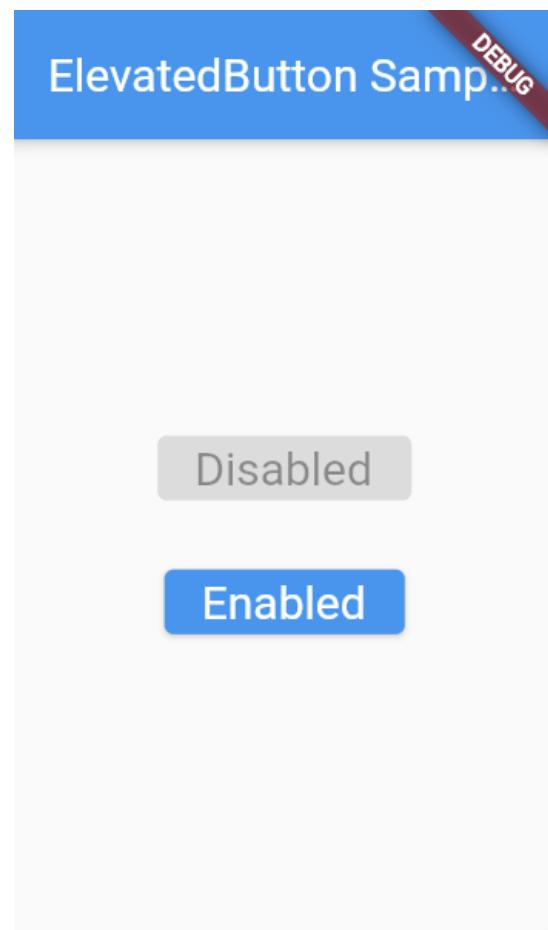
1 import 'package:flutter/material.dart';
2
3 /// Flutter code sample for [ElevatedButton].
4
5 void main() => runApp(const ElevatedButtonExampleApp());
6
7 class ElevatedButtonExampleApp extends StatelessWidget {
8   const ElevatedButtonExampleApp({super.key});

```

```
9
10    @override
11    Widget build(BuildContext context) {
12        return MaterialApp(
13            home: Scaffold(
14                appBar: AppBar(title: const Text('ElevatedButton Sample')),
15                body: const ElevatedButtonExample(),
16            ),
17        );
18    }
19}
20
21 class ElevatedButtonExample extends StatefulWidget {
22     const ElevatedButtonExample({super.key});
23
24     @override
25     State<ElevatedButtonExample> createState() => _ElevatedButtonExampleState
26     ();
27
28     class _ElevatedButtonExampleState extends State<ElevatedButtonExample> {
29         @override
30         Widget build(BuildContext context) {
31             /// Set up ElevatedButton Style
32
33             final ButtonStyle style =
34                 ElevatedButton.styleFrom(textStyle: const TextStyle(fontSize: 20));
35
36             /// Render ElevatedButton
37
38             return Center(
39                 child: Column(
40                     mainAxisSize: MainAxisSize.min,
41                     children: <Widget>[
42                         ElevatedButton(
43                             style: style,
44                             onPressed: null,
45                             child: const Text('Disabled'),
46                         ),
47                         const SizedBox(height: 30),
48                         ElevatedButton(
49                             style: style,
50                             onPressed: () {},
51                             child: const Text('Enabled'),
52                         ),
53                     ],
54                 ),
55             );
56         }
57     }
58 }
```

```
52      ) ,  
53      ] ,  
54      ) ,  
55      ) ;  
56  }  
57 }  
58 }
```

Berikut merupakan hasil layout dari code sebelumnya.



Gambar 2.5: Elevated Button

I. Flask

Flask merupakan *microframewok* yang digunakan pada sisi *backend* dengan basis bahasa pemrogramannya yaitu Phyton (Flask, 2010). Flask dirilis pada tahun

2010 dan dikembangkan oleh Armin Ronacher, seorang python *entusiast*. Flask disebut *microframework* karena Flask tidak memerlukan alat bantu lain atau *library* dalam penggunaannya.

Salah satu keuntungan menggunakan Flask adalah basis bahasa pemrogramannya menggunakan Phyton. Dengan ini, Flask dapat diintegrasikan dengan beberapa library Python seperti Machine Learning, AI, dan sebagainya.

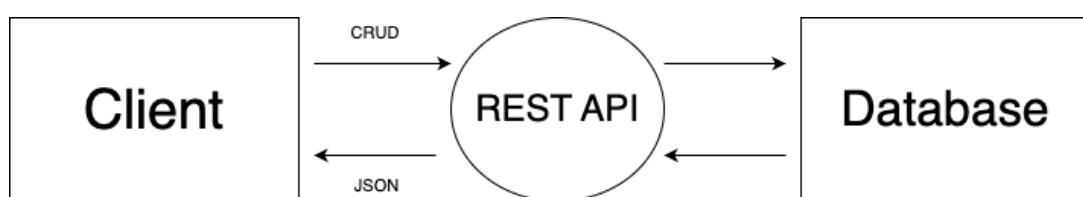
J. MongoDB

MongoDB merupakan *database* NoSQL yang dikembangkan oleh MongoDB Inc yang rilis pada tahun 2009 (MongoDB, 2009). *Database* ini disebut NoSQL karena berbasis objek atau bisa disebut JSON (JavaScript Object Notation), berbeda dengan MySQL yang berbasis tabel dalam penggunaannya.

K. REST API

Dalam pengembangan aplikasi, untuk menghubungkan antara *frontend* dengan *backend* dapat digunakan API sebagai perantaranya. API merupakan singkatan dari *Application Programming Interface* yang berfungsi menerima request dan response dari sisi *frontend* dan *backend*.

REST merupakan singkatan dari *Representational State Transfer*. API dapat disebut sebagai REST API jika memiliki standar kriteria dari REST. Kriteria tersebut bertujuan untuk menjadi standar dalam proses komunikasi antar aplikasi dan pengguna sehingga menjadi lebih fleksibel.



Gambar 2.6: Skema REST API

Pada **Gambar 2.6**, dapat dilihat skema penggunaan REST API yang

melibatkan *client* dan *database*. REST API bekerja dengan cara menerima *request* yang melibatkan *database* dan memberikan *response* kepada *client* dengan perantara komunikasi seperti HTTP.

Request client berupa CRUD (*Create, Read, Update, Delete*) pada HTTP berupa GET, POST, PUT, DELETE yang digunakan untuk berkomunikasi terhadap server serta *response* untuk *client* yang diterima berbentuk JSON. Metode GET berfungsi untuk mengambil data, POST dan PUT berfungsi untuk membuat dan memperbarui data, serta DELETE berfungsi untuk menghapus data pada *database*.

L. Scrum

Scrum merupakan metode pengembangan aplikasi yang digunakan untuk kolaborasi antar tim pengembangan (Scrum, 2010). Pada jurnal yang berjudul The Scrum Guide (Ken Schwaber, 2020), dijelaskan bahwa Scrum merupakan kerangka kerja yang digunakan untuk membantu orang, tim, serta organisasi dalam menyelesaikan suatu masalah. Scrum menerapkan prinsip Agile yang dimana berfokus pada kepuasan konsumen dalam masa pengembangan.

Untuk menjalankan metode Scrum, diperlukan Scrum Master, Product Owner, serta Developer. Scrum Master bertugas sebagai pemimpin serta bertanggung jawab dalam menjalankan prinsip Scrum pada tim, Product Owner bertanggung jawab dalam membuat dan mengontrol pekerjaan tim agar sesuai dengan kebutuhan, dan Developer yang bertugas untuk menjalankan list tugas yang sudah dibuat dan disepakati.

Dalam menjalankan Scrum, tentunya terdapat beberapa aktivitas yang digunakan agar pengembangan dapat dilakukan secara teratur. Aktivitas tersebut antara lain Sprint, Sprint Planning, Daily Scrum, Sprint Review, serta Sprint Restropective. Untuk penjelasan lebih lanjut dapat dilihat pada jurnal "The Scrum Guide" karya (Ken Schwaber, 2020) dan website Scrum yang bisa diakses pada <https://www.scrum.org/>. Adapun penelitian dari Andri Rahmanto yang menggunakan metode Scrum ini dalam pengembangan aplikasinya, penelitian

tersebut berjudul "Perancangan Arsitektur Aplikasi Budidaya Perikanan Modern pada Backend yang Bertanggung Jawab Melayani Transaksi Query Webservice Dengan Menggunakan Teknologi Flask Microservice".

M. *Unit Testing*

Unit testing merupakan suatu aktivitas dalam pengembangan aplikasi yang bertujuan untuk menguji fungsionalitas serta komponen dari aplikasi yang dikembangkan. *Unit testing* bertujuan untuk memastikan aplikasi dalam kondisi yang sudah sesuai dengan kebutuhan yang sudah disepakati sebelumnya. Proses dalam *unit testing* ini meliputi pengecekan *output* fungsi-fungsi yang ada pada aplikasi berdasarkan *input* dari *tester*.

N. *User Acceptance Test (UAT)*

Pada jurnal yang diterbitkan oleh (Hareton K.N. Leung, 1997) tentang UAT, *User Acceptance Test* (UAT) adalah tahap akhir pengujian dalam pengembangan perangkat lunak. Ketika hasil pengujian memenuhi kriteria, sistem perangkat lunak dapat dirilis untuk penggunaan operasional.

BAB III

METODOLOGI PENELITIAN

A. Inventaris Budidaya Perikanan

Inventaris pada budidaya perikanan mencakup keperluan yang digunakan selama menjalankan masa budidaya, inventaris tersebut dapat dikategorikan menjadi inventaris pakan, inventaris suplemen, inventaris listrik, inventaris benih, dan inventaris aset.

Masing-masing inventaris memiliki detail tersendiri, detail tersebut dapat dilihat dari contoh tabel inventaris berikut.

1. Inventaris Pakan

Inventaris Pakan			
<u>category</u>	Industri	Alami	Custom
<u>brand_name</u>	SNA-1	Limbah usus ayam	SNA-1 Custom
<u>description</u>	pakan protein 33%, ukuran 1 cm	sisa potong ayam yang berbentuk usus / daging	pakan protein 33% yang sudah ditingkatkan karbohidratnya
<u>min_protein</u>	33	20	30
<u>min_carbohydrate</u>	50%	-	80%
<u>min_expired_period</u>	180 hari	2 hari	4 hari
<u>max_expired_period</u>	360 hari	4 hari	8 hari
<u>amount (kg)</u>	30	2	30
<u>producer</u>	Sinta	Pedagang ayam	Pembudidaya ikan
<u>image</u>	-	-	-
<u>price</u>	Rp375,000	Rp5,000	Rp435,000

Gambar 3.1: Contoh Tabel Data Inventaris Pakan

Pada tabel ini, terdapat beberapa detail dari inventaris pakan antara lain kategori pakan, merk pakan, deskripsi pakan, protein dan karbo pakan, masa kadaluarsa pakan, jumlah pakan, produser pakan, serta harga pakan.

2. Inventaris Suplemen

Inventaris Suplemen			
_id			
function	Feed Additive	Perawatan Air	Probiotik
name	Gula	NH3 Tester	Pro Satu
description	bahan penambah karbon dengan kualitas utama	alat pengukur kadar amonia secara kimiawi	probiotik dari korea
price	15000	200000	100000
amount	1	1	1
type	kg	pack	kg
min_expired_period	360 hari	360 hari	240 hari
max_expired_period	720 hari	720 hari	540 hari
image	-	-	-

Gambar 3.2: Contoh Tabel Data Inventaris Suplemen

Pada tabel ini, terdapat beberapa detail dari inventaris suplemen antara lain fungsi suplemen, nama suplemen, deskripsi suplemen, harga suplemen, jumlah suplemen, tipe satuan suplemen, serta masa kadaluarsa suplemen.

3. Inventaris Listrik

Inventaris Listrik			
_id			
name	Token50	Token100	tagihan bulan november
price	52000	100000	70000
type	prabayar	prabayar	pascabayar
daya (kwh)	-	-	450
id_token	88991234	12345678	-
bulan	-	-	Januari
image	-	-	-

Gambar 3.3: Contoh Tabel Data Inventaris Listrik

Pada tabel ini, terdapat beberapa detail dari inventaris listrik antara lain nama, harga, tipe, daya, nomor token serta bulan bayar dari listrik.

4. Inventaris Benih

Inventaris Benih			
_id			
fish_seed_category	Benih	Benih	Pembesaran
fish_type	Nila Merah	Lele	Lele
brand_name	NilaMerah 15 gram	Lele1213	Lele50
amount	1000	1000	50
weight	-	-	1 kg
width	-	12 - 13 cm	-
price_per_unit	650	470	22000
price_total	650000	470000	1100000
image	-	-	-

Gambar 3.4: Contoh Tabel Data Inventaris Benih

Pada tabel ini, terdapat beberapa detail dari inventaris benih antara lain kategori benih, tipe ikan, nama ikan, jumlah benih, berat benih, ukuran benih, harga benih per ekor serta harga benih total.

5. Inventaris Aset

Inventaris Aset			
_id			
asset_category	Infrastruktur	Alat Budidaya	Perlengkapan Habis Pakai
name	bambu	Pacul	kertas laksus ph
description	pemasangan kolam	membuat lahan garapan	kertas laksus pengukur ph air secara kimia
amount	10	2	10
price / unit	5000	30000	10000
image	-	-	-

Gambar 3.5: Contoh Tabel Data Inventaris Aset

Pada tabel ini, terdapat beberapa detail dari inventaris aset antara lain kategori aset, nama aset, deskripsi aset, jumlah serta harga aset per unit.

B. Metode Penentuan Harga Dasar

Dalam penentuan harga dasar, penulis melakukan diskusi dengan Dinas Pertanian dan Perikanan Bogor. Berikut merupakan gambaran pada saat diskusi berlangsung.



Gambar 3.6: Diskusi dengan Dinas Pertanian dan Perikanan Bogor



Gambar 3.7: Diskusi dengan Dinas Pertanian dan Perikanan Bogor

Dari diskusi tersebut, didapatkan hasil berupa rumus penentuan harga sebagai berikut.

- Harga Jual Minimum Produk

$$T = \frac{C_p + C_q + C_r + \frac{C_l}{nA} + \frac{C_a}{60*nB}}{N} \quad (3.1)$$

Dari formula ini, dapat dihasilkan harga jual minimum produk perikanan per ekornya. Perhitungan ini didapat dari hasil diskusi antara penulis dengan klien yang merupakan pembudidaya ikan.

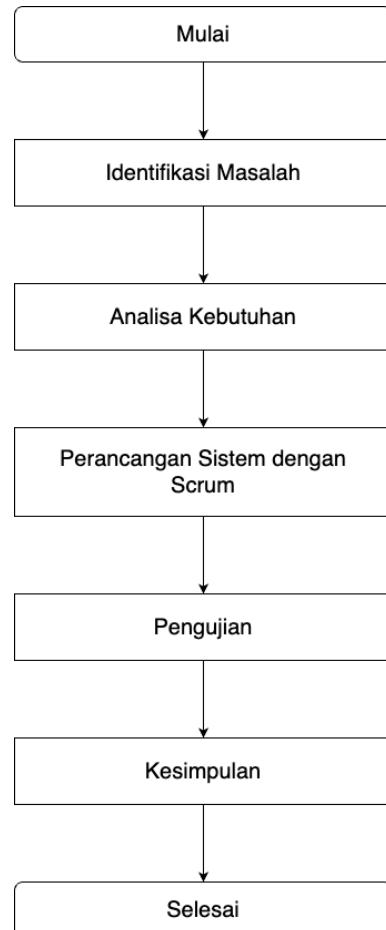
Detail atribut dari rumus dapat dilihat sebagai berikut.

- T = Harga jual minimum ikan
- C_p = Harga total pemakaian benih selama musim berjalan
- C_q = Harga total pemakaian pakan selama musim berjalan
- C_r = Harga total pemakaian suplemen selama musim berjalan
- C_l = Harga tagihan listrik per bulannya

- C_a = Harga total penggunaan aset
- nA = Jumlah kolam aktif
- nB = Jumlah semua kolam
- N = Jumlah ikan yang hidup pada kolam selama musim berjalan

Harga dari perhitungan tersebut dapat digunakan oleh pembudidaya untuk menjadi harga dasar dalam penjualannya. Tentunya harga tersebut merupakan harga panen atau produksi berdasarkan pengeluaran selama musim berjalan dan pembudidaya tidak wajib menggunakan harga tersebut dalam transaksi.

C. Tahapan Penelitian



Gambar 3.8: Alur Tahapan Penelitian

Desain penelitian adalah alur yang dijalankan selama masa pengembangan aplikasi. Pada **Gambar 3.7**, terdapat desain penelitian yang digunakan dalam pembuatan aplikasi ini dengan metode Scrum.

D. Analisa Kebutuhan

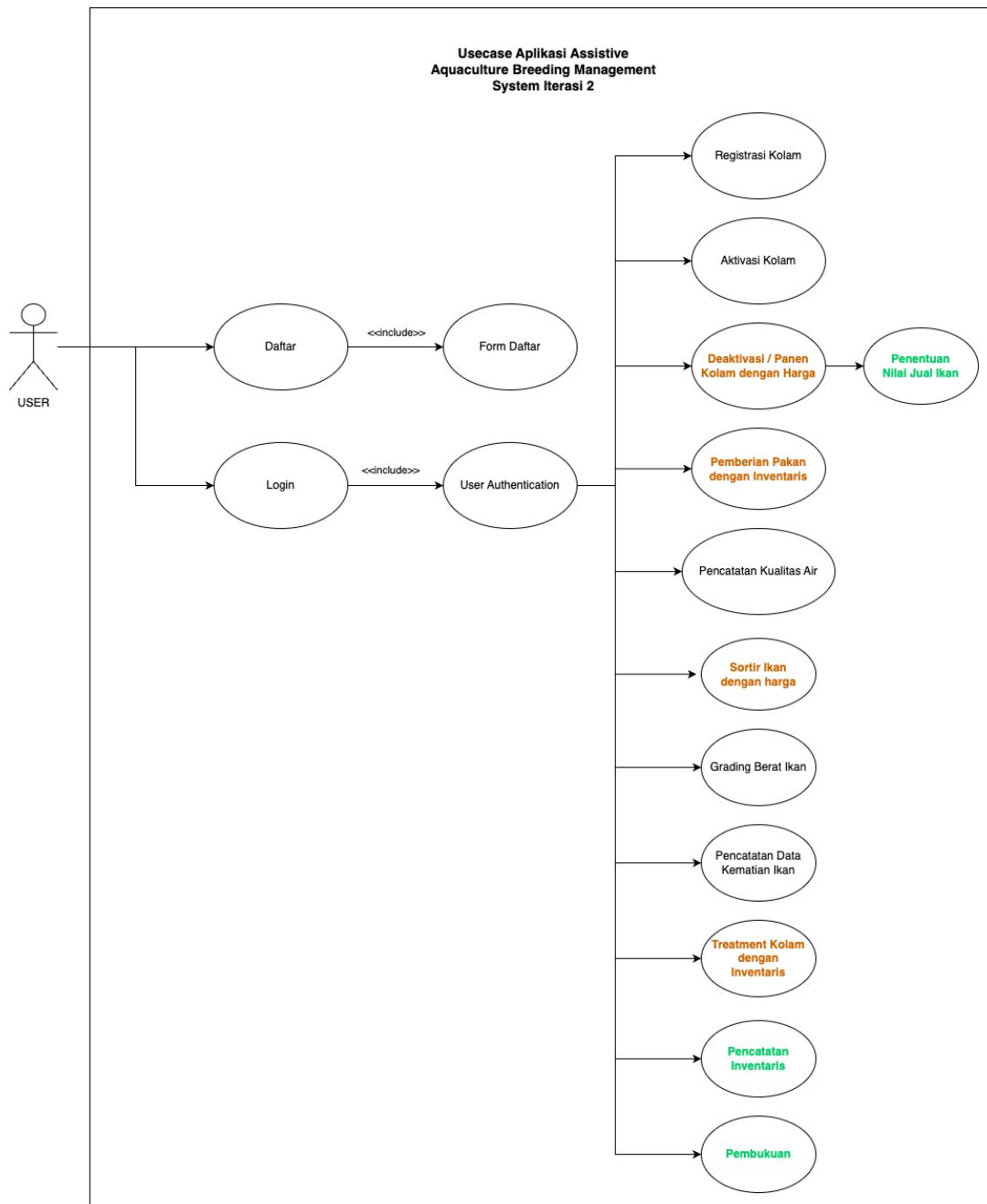
Pada pengembangan aplikasi lanjutan ini, fitur yang ditambahkan adalah fitur manajemen inventaris serta fitur pembukuan. Fitur pencatatan inventaris merupakan fitur yang akan ada pada aplikasi yang berguna untuk para pembudidaya ikan mencatat semua hal yang berhubungan dengan budidaya perikanannya. Hal-hal yang

dapat dicatat oleh pembudidaya pada fitur ini seperti bahan baku (termasuk pakan dan suplemen), penggunaan listrik, benih, serta aset yang digunakan selama masa budidaya dilakukan.

Selain mencatat inventaris pada musim budidaya, fitur pencatatan inventaris ini juga dapat menentukan rekomendasi harga jual dari ikan yang dipanen oleh pembudidaya ikan berdasarkan perhitungan dari pengeluaran biaya selama musim budidaya berjalan. Beberapa fitur yang sudah ada di penelitian sebelumnya juga harus diperbarui dengan adanya manajemen inventaris ini seperti panen, pemberian pakan, sortir ikan, serta treatment kolam.

Kemudian untuk fitur pembukuan berguna untuk pembudidaya ikan melihat riwayat musim budidaya yang sudah mereka jalankan. Terdapat beberapa rincian yang ditampilkan seperti biaya pengeluaran sampai berapa total ikan yang terpanen pada musim budidaya tersebut.

Fitur-fitur tersebut dapat dibuat menjadi *use case* pada **Gambar 3.8**. Pada *use case* tersebut, font warna hitam merupakan fitur yang sudah ada pada penelitian sebelumnya yang tidak berubah dan font warna cokelat merupakan fitur sebelumnya yang akan diperbarui pada penelitian ini. Sementara itu, untuk font warna hijau merupakan fitur baru yang akan tersedia pada aplikasi dan dikembangkan pada penelitian ini.

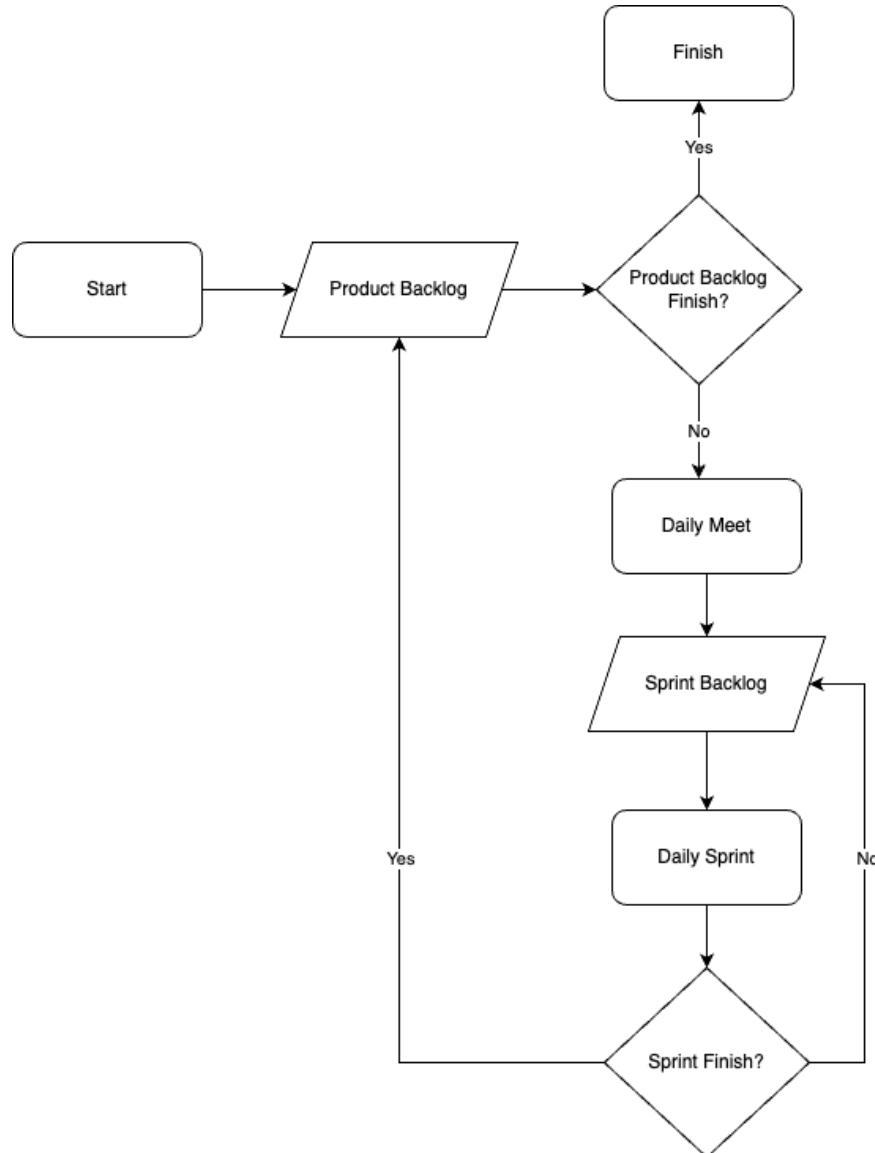


Gambar 3.9: Use Case Aplikasi

E. Perancangan Sistem

Pada aplikasi yang akan dibuat pada penelitian ini dikembangkan dengan metode Scrum. Beberapa komponen scrum seperti product backlog, sprint backlog,

daily sprint, serta daily meet digunakan agar terwujudnya ketertiban dalam masa pengembangan aplikasi. Berikut penjelasan dari masing-masing elemen yang ada pada metode Scrum.



Gambar 3.10: Tahapan Perancangan Sistem dengan Metode Scrum

1. Product Backlog

Product Backlog adalah tugas-tugas yang **akan** dijalankan pada penelitian dan hal yang pertama kali dilakukan sebelum memulai riset. Daftar tugas yang ada pada Product Backlog ini akan dipindahkan pada Sprint Backlog tergantung

pada skala prioritas dari task itu sendiri. Berikut adalah tabel dari Product Backlog yang sudah berjalan.

Tabel 3.1: Product Backlog

No	Stories	Sprint	Status
1	Fitur Pencatatan inventaris	1, 2, 3, 4, 5	Selesai
2	Fitur Aktivasi kolam dengan inventaris	3, 4, 5	Selesai
3	Fitur Pemberian pakan yang terkoneksi dengan inventaris	4	Selesai
4	Fitur Treatment kolam yang terkoneksi dengan inventaris	5	Selesai
5	Fitur Panen termasuk harga nilai jual ikan	5	Selesai
6	Fitur Pembukuan musim budidaya	5	Selesai

2. Sprint Backlog

Sprint Backlog adalah daftar tugas yang **harus** dijalankan selama masa Sprint berlangsung. Tugas yang ada pada Sprint Backlog bersifat fleksibel seiring dengan berjalannya Sprint.

3. Sprint

Progres sprint dilaksanakan ketika list task pada sprint backlog sudah disepakati bersama. Periode penggerjaan sprint bervariasi tergantung pada kesulitan task dari sprint backlog tersebut.

4. Sprint Review

Setelah Sprint berjalan, setiap minggunya diadakan meet bersama tim untuk melaksanakan Sprint Review yang bertujuan untuk melaporkan perkembangan Sprint baik itu proses ataupun hambatan selama penggerjaan Sprint.

5. Deploy Sistem

Ketika semua task sprint yang ada di sprint backlog selesai, maka aplikasi akan di deploy untuk dijalankan pengujian pada aplikasi. Pengujian aplikasi ini dilakukan dengan menggunakan Unit Testing dan User Acceptance Test (UAT).

F. Pengujian

Di tahap pengujian ini, peneliti akan melakukan uji aplikasi menggunakan dua jenis pengujian yaitu unit testing dan UAT. Pengujian unit testing dilakukan oleh tim internal developer aplikasi untuk memastikan kepastian fungsi fitur dan cara kerja fitur agar aplikasi sesuai dengan kebutuhan. Sementara UAT dilakukan agar aplikasi dapat sesuai dengan kebutuhan *user*.

1. Unit Testing

Pengujian dengan unit testing ini dibuat berdasarkan product backlog dan daftar sprint-sprint backlog yang sudah selesai. Berikut skenario pengujian yang akan dilakukan dapat dilihat pada **Tabel 3.2** berikut.

Tabel 3.2: Skenario Unit Testing

Jenis Fitur	Skenario Pengujian
Fitur Pencatatan inventaris	Saat aplikasi dibuka, terdapat tombol Kolam yang ada di footer layar
	Di halaman dashboard, terdapat tombol list yang ada pada pojok kiri atas aplikasi
	Jika tombol list ditekan, maka akan tampil beberapa list menu inventaris
	Ketika salah satu tombol pada list inventaris ditekan, maka akan masuk ke halaman detail data inventaris dari menu yang dipilih
	Pada halaman detail data inventaris, ketika tombol riwayat di pojok kanan atas ditekan akan muncul rincian input pada sistem inventaris
	Pada halaman detail data inventaris, Ketika tombol (+) yang ada di pojok kanan bawah ditekan akan masuk ke halaman input data inventaris
Fitur Aktivasi kolam dengan inventaris	Pada halaman Aktivasi Kolam, terdapat list form yang harus diisi serta pilihan benih dari inventaris yang akan digunakan pada kolam tersebut.

Jenis Fitur	Skenario Pengujian
Fitur Pemberian pakan yang terkoneksi dengan inventaris	Pada halaman Rekap Data, terdapat tombol Rekapitulasi Pakan yang akan navigasi ke halaman Rekap Pakan
	Pada halaman Rekap Pakan, terdapat tombol Entry Pakan yang akan navigasi ke halaman Entry Pakan. Disini terdapat pilihan pakan yang sebelumnya sudah dimasukkan kedalam inventaris.
Fitur Treatment kolam yang terkoneksi dengan inventaris	Pada halaman Rekap Data, terdapat tombol Treatment yang ada pada header layar yang akan navigasi ke halaman Treatment.
	Pada halaman Treatment, terdapat tombol (+) yang ada dipojok kanan bawah layar yang akan navigasi ke halaman Input Treatment.
	Pada halaman Input Treatment, terdapat list form serta pilihan suplemen yang sebelumnya sudah dimasukkan ke dalam inventaris.
Fitur Panen termasuk harga nilai jual ikan	Pada halaman Panen, terdapat list form yang diisi untuk pendataan panen serta menunjukkan harga minimum ikan.
Fitur Pembukuan pencatatan pengeluaran permusim budidaya	Pada halaman Home, terdapat tombol buku di pojok kiri atas layar yang akan navigasi ke halaman Pembukuan.
	Pada halaman Pembukuan, terdapat list data ikan hasil panen dari tiap kolam.

2. *User Acceptance Test*

User Acceptance Test dibuat berdasarkan fitur-fitur yang dapat diakses oleh pengguna pada product backlog. Berikut merupakan tabel format User Acceptance Test (UAT).

Tabel 3.3: Format *User Acceptance Test*

User Acceptance Test					
No	Acceptance Requirements	Kesesuaian			
		SS	S	TS	STS
1	Fitur inventaris pakan				
2	Fitur inventaris suplemen				
3	Fitur inventaris benih				
4	Fitur inventaris listrik				
5	Fitur inventaris aset				
6	Aktivasi kolam dengan inventaris				
7	Pemberian pakan dengan inventaris				
8	Treatment kolam dengan inventaris				
9	Panen kolam dengan harga jual minimum ikan				
10	Pembukuan panen musim budidaya				

BAB IV

HASIL DAN PEMBAHASAN

A. Perancangan Sistem Dengan Scrum

Pengembangan aplikasi Aqua Breeding ini dirancang menggunakan metode Scrum. Dalam metode Scrum, proses pengembangan dilakukan secara bertahap yang bisa disebut sebagai Sprint. Pada penelitian ini terdapat 5 Sprint yang masing-masing Sprintnya memiliki waktu penyelesaian yang berbeda-beda. Sebelum memulai Sprint, dilakukan perencanaan Sprint Backlog yang diambil dari Product Backlog yang sebelumnya sudah disepakati. Adapun laporan setiap Sprint dalam proses pengembangan sistem dapat dilihat sebagai berikut :

1. Sprint 1

Sprint 1 dilaksanakan pada tanggal 07 Maret 2023 - 29 Maret 2023. Detail dari Sprint 1 ini adalah mengerjakan tugas yang ada pada Sprint 1 Backlog di tabel berikut.

Tabel 4.1: Sprint 1 Backlog

No	Stories	Task	Status
1	Fitur pencatatan inventaris	- Membuat skema database dari pencatatan inventaris - Membuat integrasi skema database dengan skema database sebelumnya - Membuat mockup dari fitur inventaris	Selesai Selesai Selesai

Berikut merupakan skema database yang mewakili fitur inventaris dapat dilihat pada **Gambar 4.1** berikut.

fish_feed		
PK	id	ObjectId()
	name	String NOT NULL
	price	Int NOT NULL
	amount	Int NOT NULL
	type	String NOT NULL
	protein	String NOT NULL
	carbohydrate	String NOT NULL
	expired_period	Int NOT NULL
	created_at	date NOT NULL
	updated_at	date NOT NULL

fish_seed		
PK	id	ObjectId()
	fish_type	String NOT NULL
	amount	Int NOT NULL
	weight	Int NOT NULL
	price	Int NOT NULL
	created_at	date NOT NULL
	updated_at	date NOT NULL

material_organic		
PK	id	ObjectId()
	name	String NOT NULL
	price	Int NOT NULL
	amount	Int NOT NULL
	type	String NOT NULL
	expired_period	Int NOT NULL
	created_at	date NOT NULL
	updated_at	date NOT NULL

asset_tools		
PK	id	ObjectId()
	name	String NOT NULL
	function	String NOT NULL
	price	Int NOT NULL
	created_at	date NOT NULL
	updated_at	date NOT NULL

electric_bill		
PK	id	ObjectId()
	amount	Int NOT NULL
	price	Int NOT NULL
	created_at	date NOT NULL
	updated_at	date NOT NULL

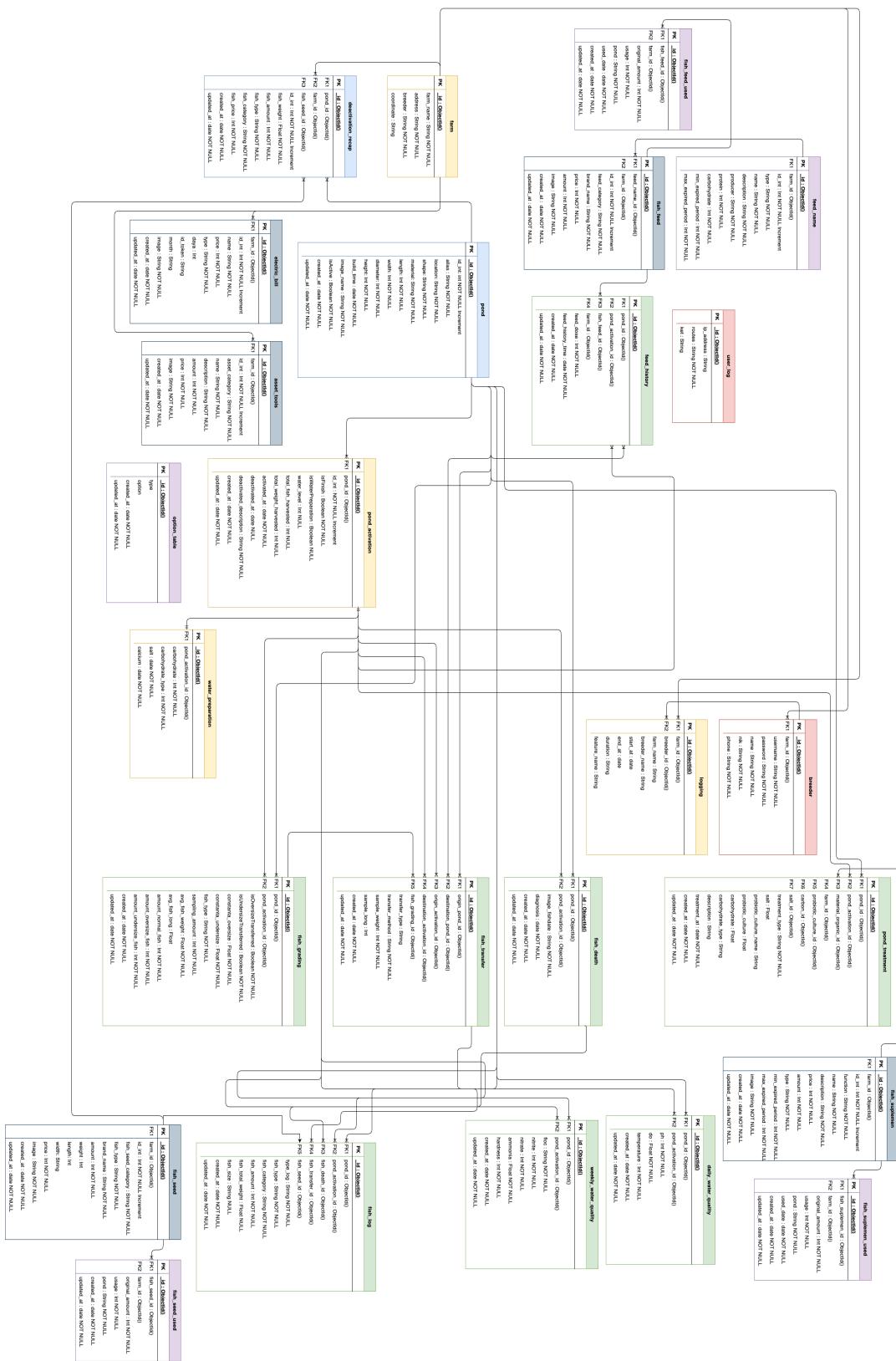
Gambar 4.1: Skema Database Fitur Inventaris

Dari skema database tersebut, terdapat lima opsi kategori inventaris yang sudah dijelaskan sebelumnya. Pada skema database ini, masing-masing kategori memiliki kebutuhan yang berbeda antara lain.

1. fish_feed (Pakan Ikan)
2. material_organic (Bahan Organik)
3. electric_bill (Tagihan Listrik)
4. fish_seed (Benih Ikan)
5. asset_tools (Peralatan)

Dalam tabel database tersebut, pada kolom pertama terdapat jenis *key* yang dijadikan patokan dalam tabel database tersebut. Kemudian kolom kedua dan ketiga merupakan hubungan antara nama data dan tipe data yang mewakili nama data tersebut.

Setelah skema database dari inventaris telah dibuat, tabel-tabel database tersebut harus diintegrasikan dengan skema database sebelumnya untuk menyesuaikan kebutuhan fitur yang akan dibuat nantinya. Berikut merupakan skema database yang telah diintegrasikan dengan skema database dari inventaris dapat dilihat pada **Gambar 4.2**.



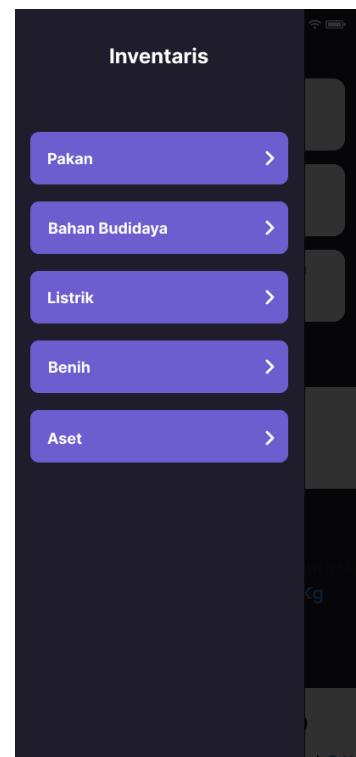
Gambar 4.2: Integrasi Database Inventaris dengan Skema Database Iterasi 1

Dari integrasi diatas, tabel fish_feed diintegrasikan dengan tabel feed_type yang digunakan untuk input pakan dan tabel material_organic diintegrasikan dengan tabel pond_treatment karena dalam fitur treatment kolam diperlukan data dari tabel material organik tersebut. Lalu tabel fish_seed diintegrasikan dengan tabel pond_activation dan fish_log untuk aktivasi kolam dan perhitungan jumlah ikan. Sementara itu, tabel electric_bill dan asset_tools merupakan individu yang tidak terintegrasi dengan tabel yang lain. Hal ini dikarenakan tabel tersebut hanya untuk menyimpan datanya saja dan tidak digunakan di dalam fitur.

Berikut merupakan mockup dari fitur inventaris yang mencakup skema database sebelumnya.



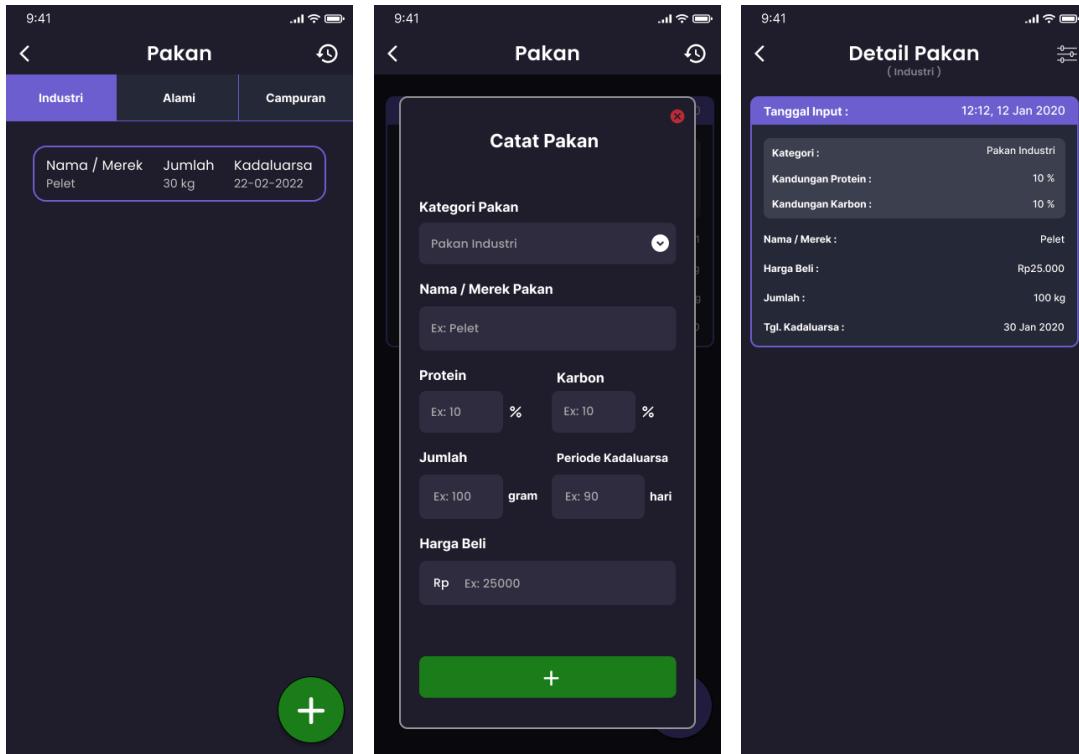
Gambar 4.3: Halaman Dashboard



Gambar 4.4: Halaman Menu Inventaris

Pada halaman dashboard, di pojok kiri atas terdapat ikon *hamburger* atau list yang ketika ditekan akan menampilkan halaman menu inventaris seperti **Gambar 4.4**. Masing-masing list menu yang ada pada halaman menu inventaris memiliki fungsi

yang sesuai dengan skema inventaris.



Gambar 4.5: Halaman Data Inventaris Pakan

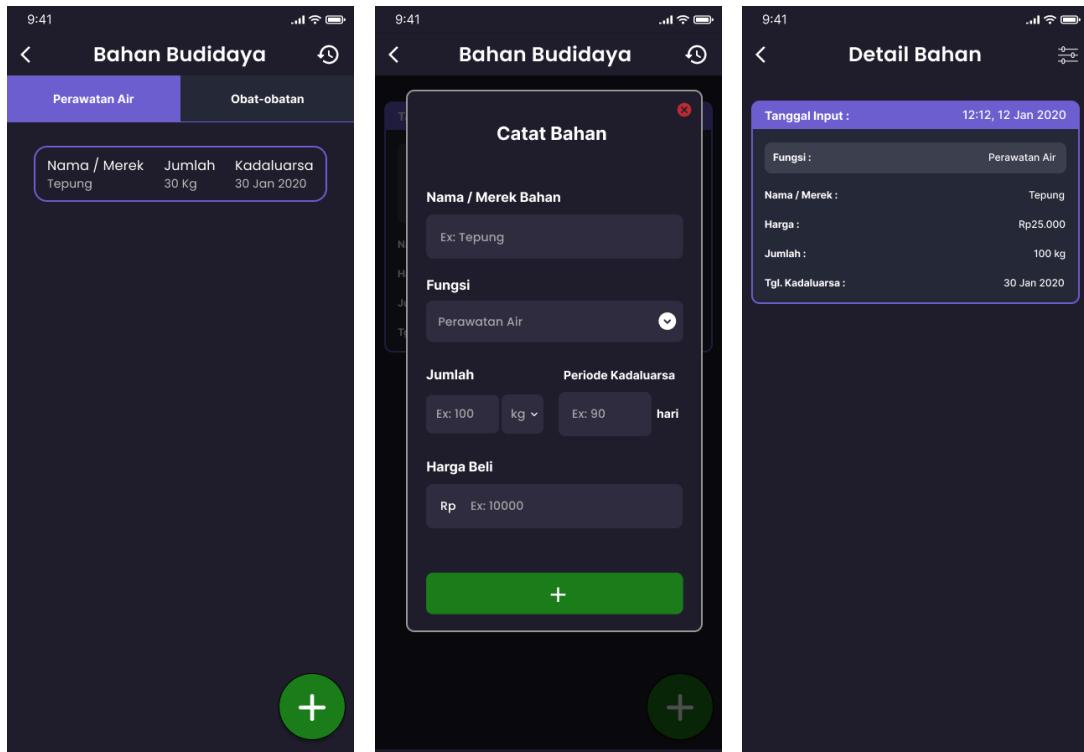
Gambar 4.6: Halaman Input Inventaris Pakan

Gambar 4.7: Halaman Detail Inventaris Pakan

Jika pada halaman menu inventaris sebelumnya dipilih menu "Pakan", maka akan masuk ke halaman data inventaris pakan. Pada halaman ini terdapat 3 jenis pakan yaitu pakan industri (pelet), alami (tumbuh-tumbuhan), serta campuran (tepung, terigu, dll). Masing-masing jenis pakan memiliki detail data yaitu nama atau merek pakan, total jumlah pakan yang tersedia, serta tanggal kadaluarsa dari pakan tersebut.

Tombol (+) yang ada di pojok kanan bawah akan mengarahkan ke halaman input dari inventaris pakan. Disini diberikan form input yang beragam seperti yang ada pada **Gambar 4.6**.

Sementara tombol riwayat yang ada di pojok kanan atas akan mengarahkan ke halaman detail dari inventaris pakan. Di halaman ini ditampilkan detail pemasukkan pakan ke sistem inventaris.



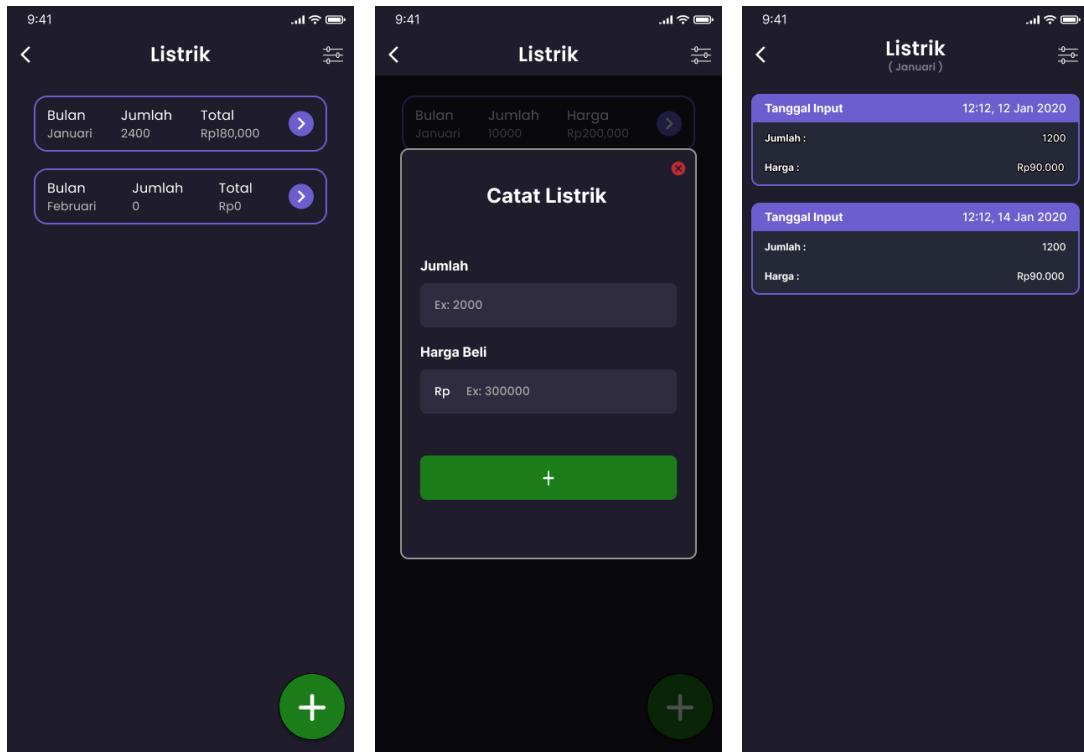
Gambar 4.8: Halaman Data Inventaris Bahan Budidaya

Gambar 4.9: Halaman Input Inventaris Bahan Budidaya

Gambar 4.10: Halaman Detail Inventaris Bahan Budidaya

Jika pada halaman menu inventaris sebelumnya dipilih menu "Bahan Budidaya", maka akan masuk ke halaman data inventaris bahan budidaya. Pada halaman ini terdapat 2 jenis bahan budidaya yang dibagi berdasarkan fungsi yaitu perawatan air dan obat-obatan (Methylene Blue, dll). Sama seperti pada inventaris pakan, masing-masing jenis bahan budidaya memiliki detail data yaitu nama atau merek, total jumlah yang tersedia, serta tanggal kadaluarsa.

Sama seperti halaman inventaris pakan, tombol (+) mengarahkan ke halaman input seperti **Gambar 4.9** dan tombol riwayat akan mengarahkan ke halaman detail inventaris seperti pada **Gambar 4.10**.



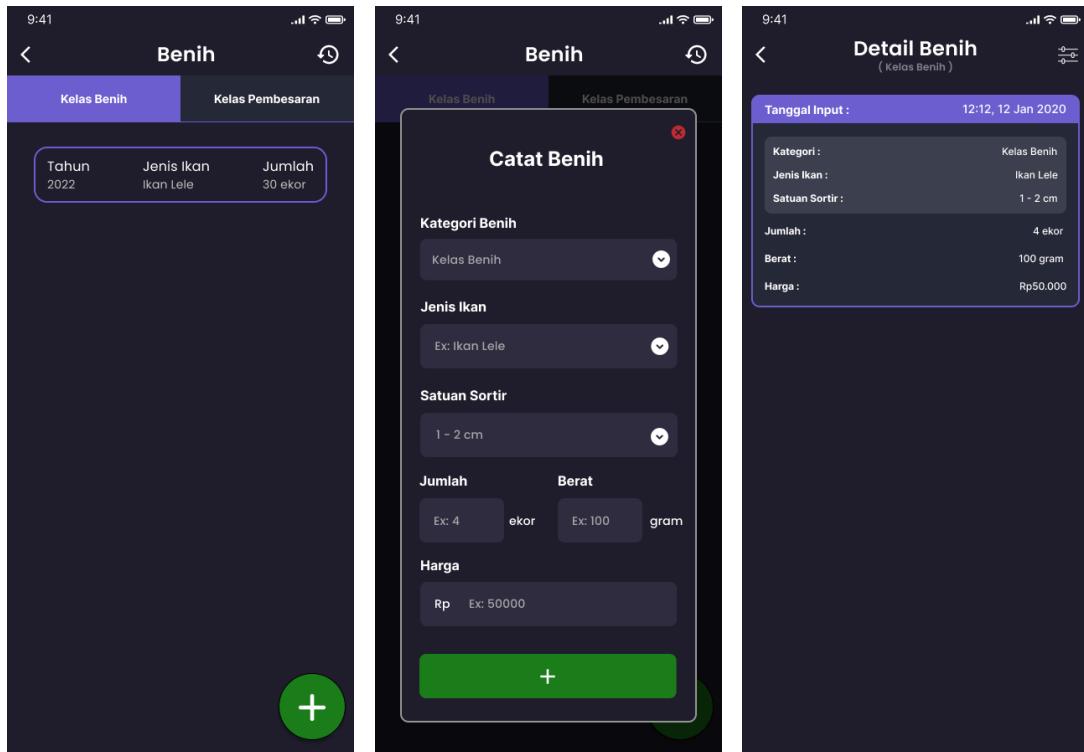
Gambar 4.11: Halaman Data Inventaris Tagihan Listrik

Gambar 4.12: Halaman Input Inventaris Tagihan Listrik

Gambar 4.13: Halaman Detail Inventaris Tagihan Listrik

Jika pada halaman menu inventaris sebelumnya dipilih menu "Listrik", maka akan masuk ke halaman data inventaris tagihan listrik. Pada halaman ini terdapat list dari tagihan listrik perbulan yang digunakan oleh pembudidaya, data yang ditampilkan berupa bulan, jumlah listrik, serta total biaya tagihan. Jika list bulan tersebut ditekan, maka akan pindah ke halaman detail dari tagihan listrik dibulan tersebut.

Untuk tombol (+) yang ada di pojok kanan bawah, jika ditekan akan masuk ke halaman input tagihan. Form yang harus diisi hanya jumlah token listrik dan harga belinya. Sementara tombol filter yang ada di pojok kanan atas berfungsi untuk memfilter data sesuai keinginan user.



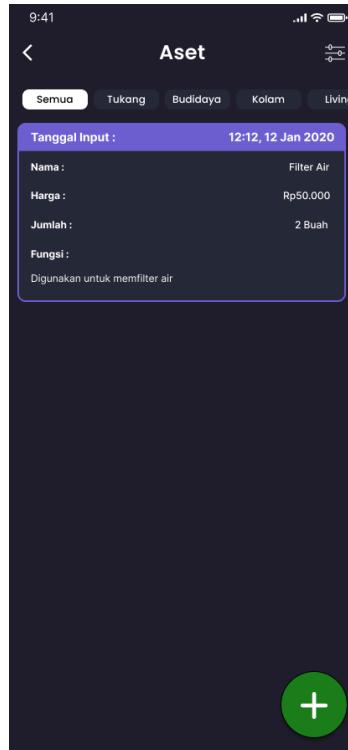
Gambar 4.14: Halaman Data Inventaris Benih

Gambar 4.15: Halaman Input Inventaris Benih

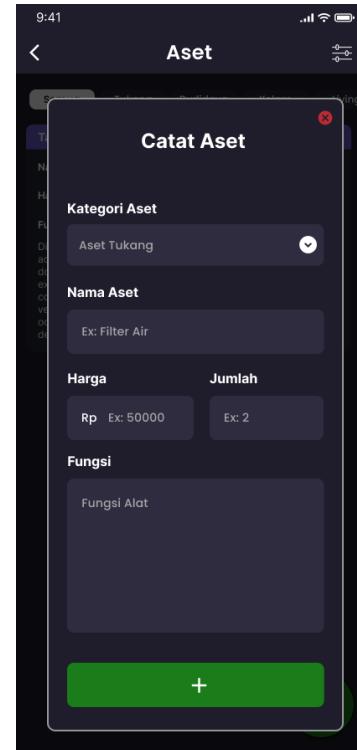
Gambar 4.16: Halaman Detail Inventaris Benih

Jika pada halaman menu inventaris sebelumnya dipilih menu "Benih", maka akan masuk ke halaman data inventaris benih. Pada halaman ini terdapat list dari benih yang sudah diinput pada sistem inventaris yang terbagi menjadi dua jenis yaitu benih jenis kelas benih (kecil) dan kelas pembesaran (besar). Masing-masing data memiliki detail seperti tahun benih di input, jenis benih, dan jumlah dari benih.

Untuk tombol (+) yang ada di pojok kanan bawah, jika ditekan akan dinavigasikan ke halaman input benih ikan. Halaman input ini memiliki form seperti pada **Gambar 4.15**. Terdapat dua jenis form yang berbeda berdasarkan kategori yang dipilih, untuk kategori kelas benih pada bagian ukurannya menggunakan satuan sortir sementara untuk kategori kelas pembesaran menggunakan panjang dan lebar untuk ukurannya.



Gambar 4.17: Halaman Data Inventaris Aset



Gambar 4.18: Halaman Input Inventaris Aset

Jika pada halaman menu inventaris sebelumnya dipilih menu "Aset", maka akan masuk ke halaman data inventaris aset. Pada halaman ini, ditampilkan jenis dari aset-aset yang digunakan selama masa budidaya.

Aset dibagi menjadi empat jenis kategori yaitu aset tukang (aset yang diperlukan pembudidaya), aset budidaya (aset yang dibutuhkan selama budidaya berlangsung), aset kolam (aset yang digunakan dalam kolam budidaya), dan aset living (aset yang diperlukan selama berlangsungnya musim budidaya).

Tombol (+) pada pojok kanan bawah berfungsi untuk navigasi ke halaman input sementara tombol filter pada pojok kanan atas berfungsi untuk filter data.

2. Sprint 2

Sprint 2 dilaksanakan pada tanggal 30 Maret 2023 - 15 April 2023. Detail dari Sprint 2 ini adalah mengerjakan tugas yang ada pada Sprint 2 Backlog di tabel

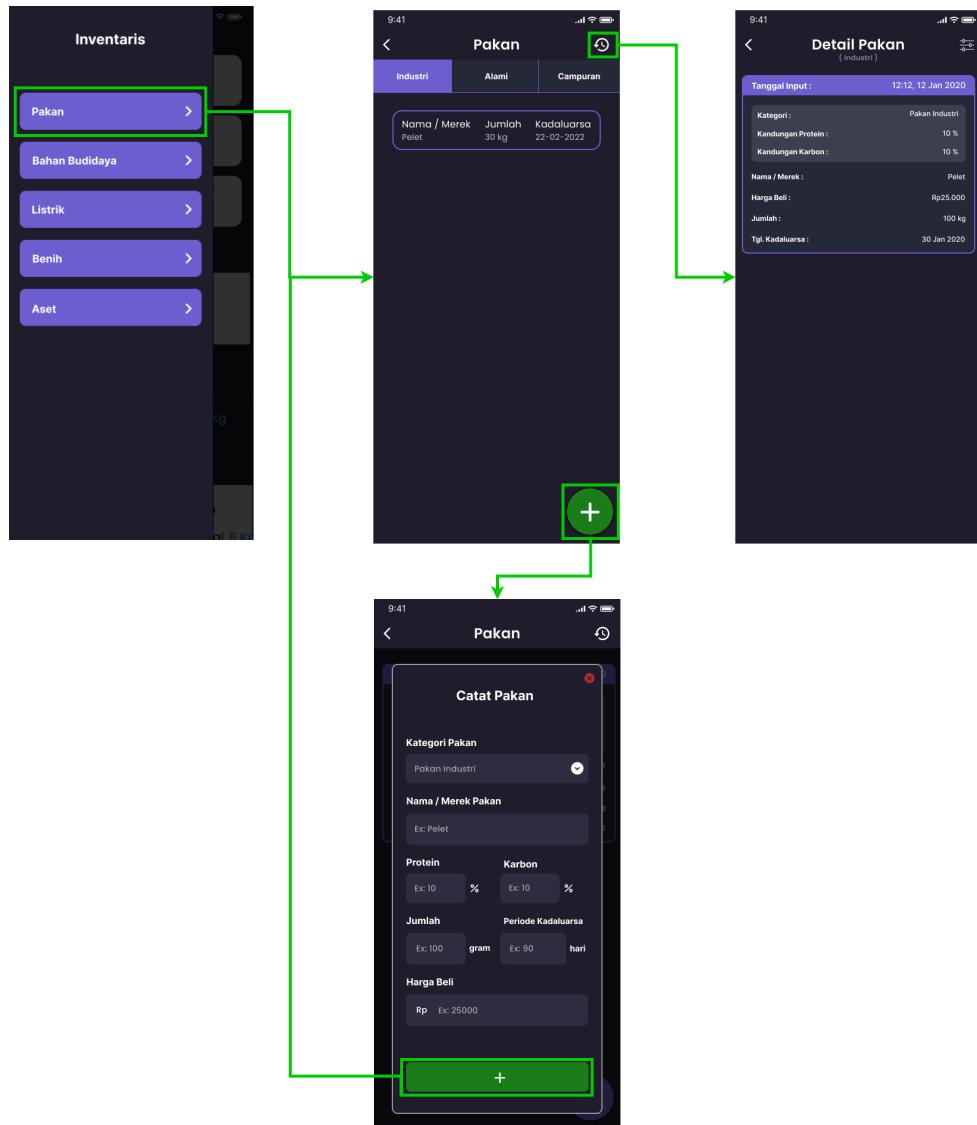
berikut.

Tabel 4.2: Sprint 2 Backlog

No	Stories	Task	Status
1	Fitur pencatatan inventaris	<ul style="list-style-type: none"> - Membuat alur UI/UX dari design aplikasi - Mengupdate skema database pada inventaris 	Selesai Selesai

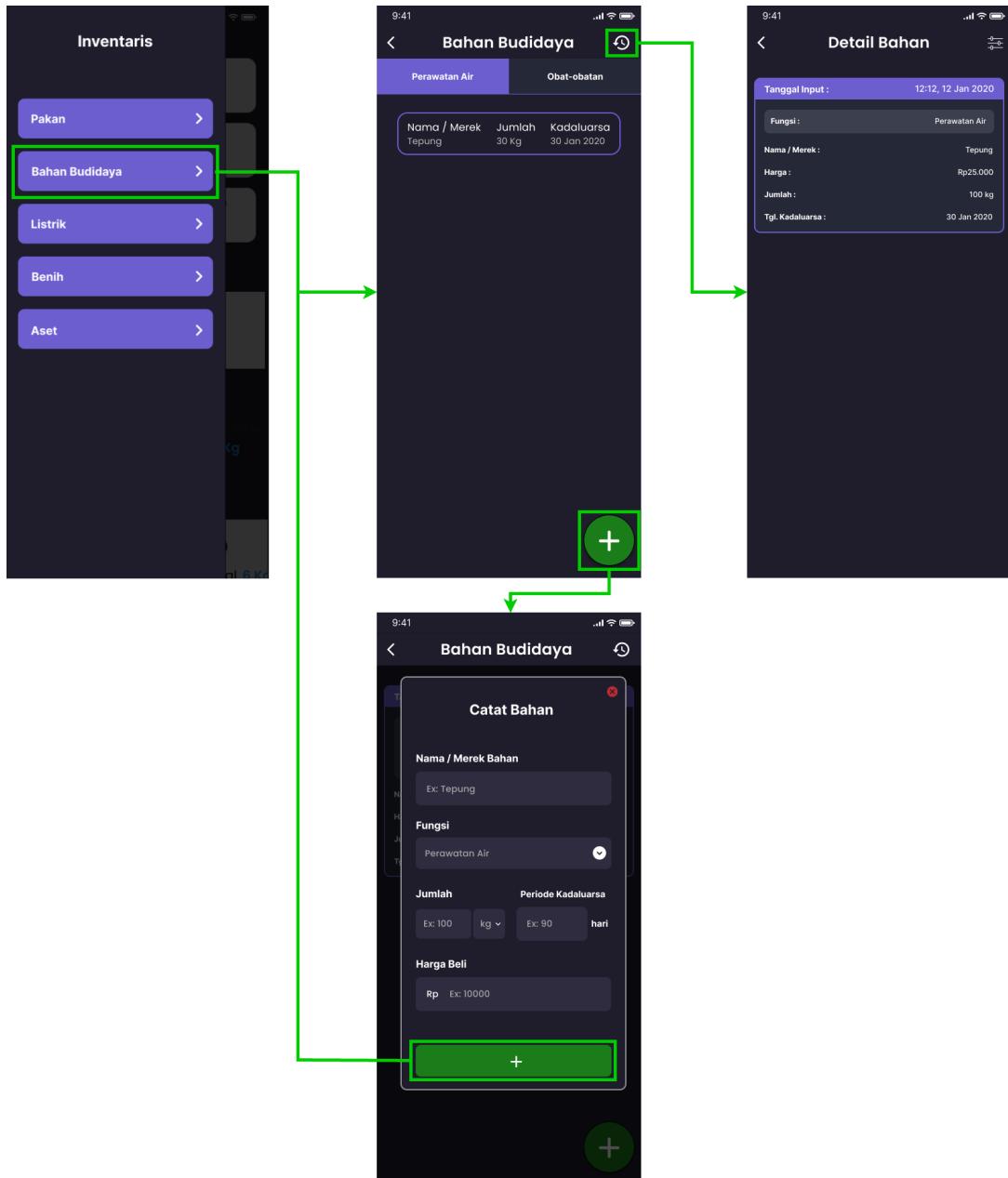
Selama masa Sprint 2 berlangsung, tim bertemu dengan perwakilan dari Dinas Perikanan Bogor. Dari pertemuan itu, salah satunya terdapat beberapa perubahan user requirement sehingga perlu merubah skema database yang sudah dibuat pada Sprint 1. Lalu, terdapat juga pertimbangan perubahan penentuan harga jual ikan dengan ditambahkannya perhitungan aset namun hal ini belum ditentukan akan masuk perhitungan atau tidak.

Berikut merupakan alur user sebagai pengguna aplikasi berdasarkan mockup yang sudah dibuat pada Sprint 1.



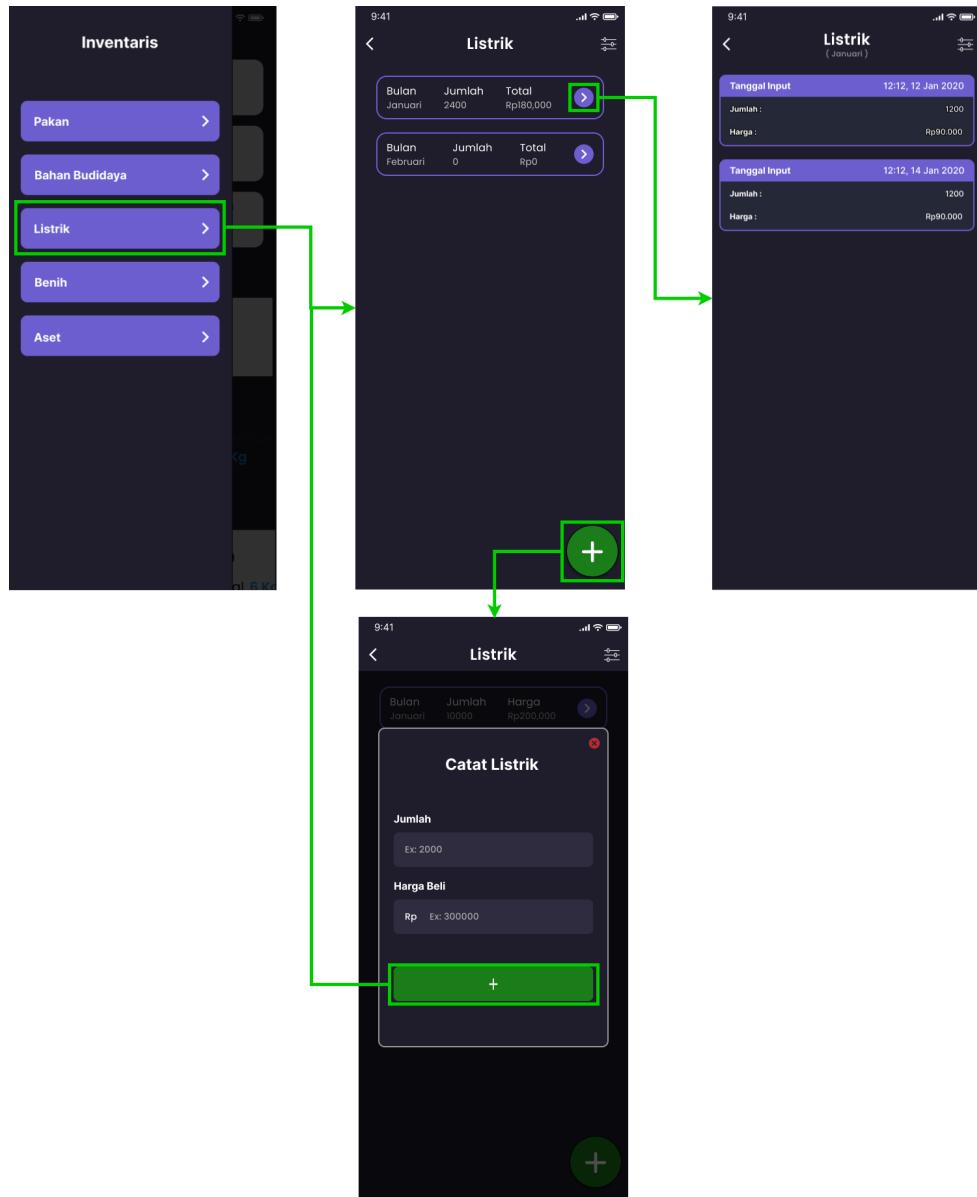
Gambar 4.19: Alur Inventaris Pakan

Pada **Gambar 4.19** merupakan alur dari inventaris pakan. Pengguna akan memilih menu "Pakan" dan masuk ke halaman data inventaris pakan, kemudian tombol (+) akan navigasikan pengguna ke halaman input pakan dan tombol riwayat akan navigasikan pengguna ke halaman detail input pakan.



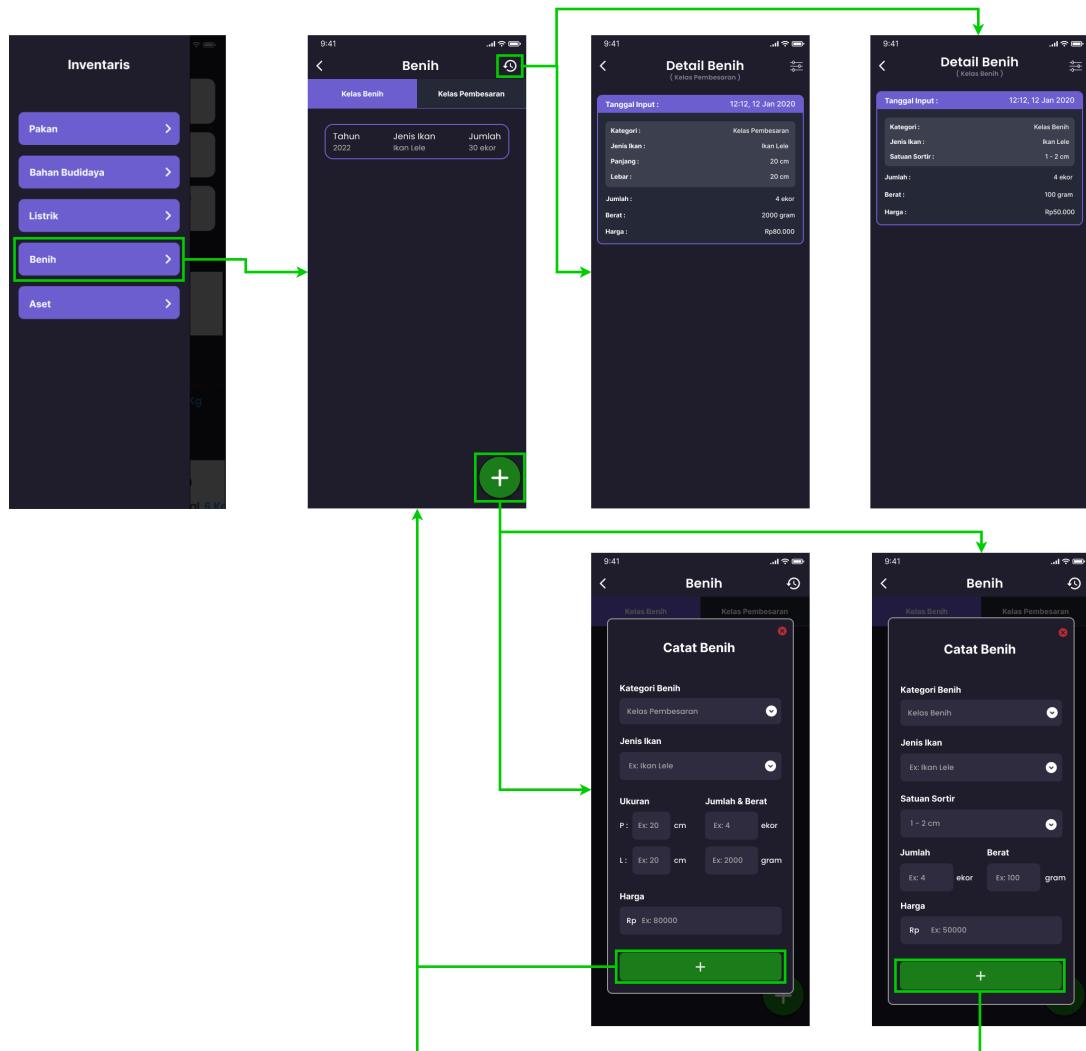
Gambar 4.20: Alur Inventaris Bahan Budidaya

Pada **Gambar 4.20** merupakan alur dari inventaris bahan budidaya. Pengguna akan memilih menu "Bahan Budidaya" dan masuk ke halaman data inventaris budidaya, kemudian tombol (+) akan navigasikan pengguna ke halaman input bahan budidaya dan tombol riwayat akan navigasikan pengguna ke halaman detail input bahan budidaya.



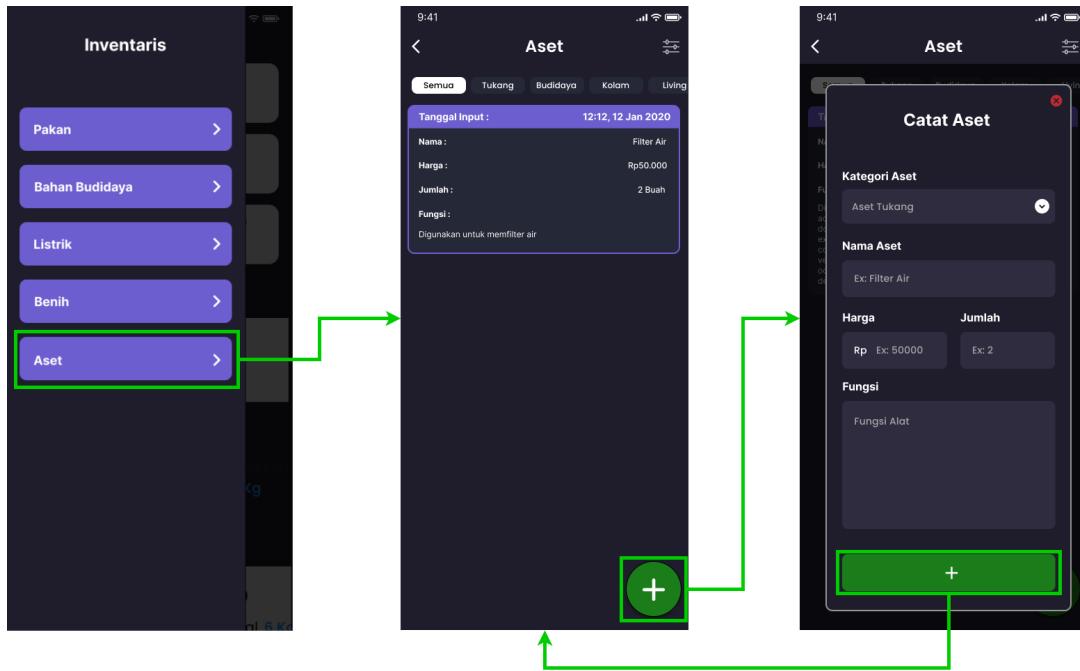
Gambar 4.21: Alur Inventaris Listrik

Pada **Gambar 4.21** merupakan alur dari inventaris listrik. Pengguna akan memilih menu "Listrik" dan masuk ke halaman data inventaris listrik, kemudian tombol (+) akan navigasikan pengguna ke halaman input listrik serta jika pengguna menekan salah satu list bulan pada data listrik, maka akan masuk ke halaman detail input tagihan listrik.



Gambar 4.22: Alur Inventaris Benih

Pada **Gambar 4.22** merupakan alur dari inventaris benih. Pengguna akan memilih menu "Benih" dan masuk ke halaman data inventaris benih, kemudian tombol (+) akan menavigasikan pengguna ke halaman input benih dan tombol riwayat akan menavigasikan pengguna ke halaman detail input benih.



Gambar 4.23: Alur Inventaris Aset

Pada **Gambar 4.23** merupakan alur dari inventaris aset. Pengguna akan memilih menu "Aset" dan masuk ke halaman data inventaris aset, kemudian tombol (+) akan navigasikan pengguna ke halaman input aset.

Setelah semua alur telah selesai dibuat, selanjutnya merupakan perubahan skema database inventaris yang dapat dilihat pada **Gambar 4.24** berikut.

fish_feed		
PK	id feed_category_id	ObjectId() Int NOT NULL String NOT NULL Int NOT NULL Int NOT NULL Int NOT NULL Int NOT NULL date NOT NULL date NOT NULL
	name price amount protein carbon expired_period created_at updated_at	

fish_seed		
PK	id fish_seed_category_id	ObjectId() Int NOT NULL String NOT NULL Int NOT NULL Int NOT NULL String NULL Int NULL Int NULL Int NOT NULL date NOT NULL date NOT NULL
	fish_type amount weight sorting_size length width price created_at updated_at	

material_parts		
PK	id function_id	ObjectId() Int NOT NULL String NOT NULL Int NOT NULL Int NOT NULL String NOT NULL Int NOT NULL date NOT NULL date NOT NULL
	name price amount type expired_period created_at updated_at	

asset_tools		
PK	id asset_category_id	ObjectId() Int NOT NULL String NOT NULL String NOT NULL Int NOT NULL Int NOT NULL date NOT NULL date NOT NULL
	name function amount price created_at updated_at	

electric_bill		
PK	id amount	ObjectId() Int NOT NULL Int NOT NULL date NOT NULL date NOT NULL
	price created_at updated_at	

Gambar 4.24: Update Skema Database Inventaris

Berdasarkan skema database inventaris tersebut, jika dibandingkan dengan skema database inventaris sebelumnya pada **Gambar 4.1** terdapat pembaruan pada bagian inventaris pakan, bahan budidaya, benih, dan aset.

Pada skema database di inventaris pakan (fish_feed), ditambahkan *key feed_category_id* karena pada inventaris pakan diharuskan memilih kategori pakan yang akan dimasukkan. Untuk itu feed_category_id berperan untuk menampung jenis kategori pakan yang akan diinput. Sebelumnya jenis inventaris pakan tidak memiliki kategori, sehingga perlu ditambahkan *key* baru untuk jenis data kategori tersebut.

Kemudian, di skema database inventaris bahan budidaya (material_parts) ditambahkan *key function_id* karena bahan budidaya dibagi menjadi dua fungsi yaitu perawatan air dan obat-obatan. Sebelumnya inventaris bahan budidaya tidak memiliki kategori, sehingga harus ditambahkan *key* baru pada database untuk

menampung jenis data tersebut.

Lalu pada skema database inventaris benih ditambahkan *key fish_seed_category_id, sorting_size, length*, serta **width**. *key* tersebut ditambahkan karena pada benih dibagi menjadi dua jenis yaitu kelas benih dan kelas pembesaran. Masing-masing kategori memiliki jenis data pengukuran yang berbeda, pada kelas benih digunakan pengukuran satuan sortir dengan *key sorting_size* sementara kelas pembesaran digunakan pengukuran panjang dan lebar dengan *key length* dan *width*. Sebelumnya untuk benih tidak terdapat kategori dan jenis ukuran benih sehingga *key* baru diperlukan untuk menampung data tersebut.

Terakhir terdapat perubahan pada skema database inventaris aset yaitu ditambahkan *key asset_category_id* dan **amount**. Sebelumnya inventaris aset hanya menampung segala jenis aset yang digunakan pada musim budidaya tanpa adanya jenis kategori dan jumlah yang spesifik, namun di skema database sekarang dapat ditentukan jenis kategori pada aset dan berapa jumlah aset yang digunakan sehingga pemantauan aset yang digunakan menjadi lebih detail.

3. Sprint 3

Sprint 3 dilaksanakan pada tanggal 10 Mei 2023 - 21 Juni 2023. Detail dari Sprint 3 ini adalah mengerjakan tugas yang ada pada Sprint 3 Backlog di tabel berikut.

Tabel 4.3: Sprint 3 Backlog

No	Stories	Task	Status
1	Fitur pencatatan inventaris	<ul style="list-style-type: none"> - Design route untuk inventaris benih (dalam bentuk RESTful API) - Membuat halaman serta Integrasi RESTful API benih dengan Flutter 	Selesai
2	Fitur Aktivasi kolam dengan inventaris	<ul style="list-style-type: none"> - Membuat tabel riwayat pemakaian benih - Design route dan penerapan dengan Flutter untuk riwayat pemakaian benih (dalam bentuk RESTful API) 	Selesai Selesai

Dalam proses Sprint ini, beberapa tahapan yang dilakukan adalah sebagai berikut :

1. Design route untuk inventaris benih (dalam bentuk RESTful API)

Pada tugas ini, langkah pertama yang dilakukan adalah membuat sample route berupa url endpoint yang nantinya akan digunakan Flutter untuk berkomunikasi dengan backend. Hasilnya adalah sebagai berikut:

Group	Endpoint	HTTP Status	Operation	Purpose
Inventaris Benih	/inventory/seed?type=	GET	READ	mengambil semua data inventaris benih dengan filter type
	/inventory/seed/\${id}	GET	READ	mengambil data spesifik dari inventaris benih dengan id
	/inventory/seed	POST	CREATE	menginput data kedalam inventaris benih
	/inventory/seed/\${id}	PUT	UPDATE	memperbarui spesifik data pada inventaris benih dengan id
	/inventory/seed/\${id}	DELETE	DELETE	menghapus spesifik data pada inventaris benih dengan id

Gambar 4.25: Sample Route Benih

Kemudian, sebelum mengimplementasikan route tersebut dalam backend. Hal yang harus dilakukan adalah membuat model database dari inventaris benih

berdasarkan skema database yang sudah dibuat pada Sprint 2. Berikut merupakan model inventaris benih pada Flask:

- Model Inventaris Benih

```

1      class SeedInventory(db.Document):
2          id_int = db.SequenceField(required=True)
3          farm_id = db.ReferenceField(Farm, required=True)
4          fish_seed_category = db.StringField(required=True)
5          fish_type = db.StringField(required=True)
6          brand_name = db.StringField(required=True)
7          amount = db.IntField(required=True)
8          weight = db.FloatField(default=0)
9          width = db.StringField()
10         price = db.IntField(required=True)
11         total_price = db.IntField(required=True)
12         image = db.StringField(required=True)
13         created_at = db.DateTimeField(default=datetime.datetime.now)
14         updated_at = db.DateTimeField(default=datetime.datetime.now)
15

```

Setelah model dibuat, pengimplementasian model tersebut dalam Flask diperlukan bentuk class yang isinya mengandung fungsi-fungsi HTTP Request yang diperlukan. Berikut class hasil pengimplementasian model yang sudah dibuat sebelumnya sebagai berikut :

(a) Mengambil semua data benih (HTTP Method - GET)

Fungsi ini digunakan untuk menerima GET request dan akan merespon dengan data yang ada pada database inventaris benih.

```

1      class SeedInventoriesApi(Resource):
2          @jwt_required()
3          def get(self):
4              try:
5                  current_user = get_jwt_identity()
6                  farm = str(current_user['farm_id'])
7                  farm_id = ObjectId(farm)
8
9                  type = request.args.get('type') if request.args.get('type')
else ""

```

```

10
11     pipeline = [
12         {"$sort": {"id_int": 1}},
13         {
14             '$match': {
15                 "farm_id": farm_id,
16                 'fish_seed_category': {
17                     '$regex': type,
18                     '$options': 'i'
19                 }
20             }
21         },
22     ]
23
24     testing = SeedInventory.objects.aggregate(pipeline)
25     temp = list(testing)
26     response = json.dumps({
27         'status': 'success',
28         'data': temp,
29     }, default=str)
30     return Response(response, mimetype="application/json",
31     status=200)
32     except Exception as e:
33         response = {"message": e}
34         response = json.dumps(response, default=str)
35         return Response(response, mimetype="application/json",
36         status=400)

```

Di fungsi tersebut terdapat variabel type yang berisi request.args.get yang digunakan untuk mengambil data parameter sebagai filter kepada database yang diambil. Pipeline pada fungsi ini berguna untuk menentukan secara spesifik bagaimana bentuk data yang akan diambil nanti dari database inventaris benih.

Jika tidak ada kendala dalam request data maka akan mendapatkan response status 200 yang berarti berhasil, jika tidak akan mendapatkan response status 400 yang berarti ada masalah saat melakukan request.

(b) Mengambil data benih secara spesifik berdasarkan ID benih (HTTP Method - GET)

Fungsi ini digunakan untuk menerima GET request dan akan merespon dengan data yang ada pada database inventaris benih secara spesifik sesuai dengan ID benih.

```

1      class SeedInventoryApi(Resource):
2          def get(self, id):
3              try:
4                  pipeline = {"$match": {"id_int": int(id)}},
5                  testing = SeedInventory.objects.aggregate(pipeline)
6                  temp = list(testing)
7                  if len(temp) == 0:
8                      res = {"message": 'no data found'}
9                      response = json.dumps(res, default=str)
10                     return Response(response, mimetype="application/json",
11                                     status=200)
12                     response = json.dumps({
13                         'status': 'success',
14                         'data': temp[0],
15                         }, default=str)
16                     return Response(response, mimetype="application/json",
17                                     status=200)
18                     except Exception as e:
19                         response = {"message": e}
20                         response = json.dumps(response, default=str)
21                         return Response(response, mimetype="application/json",
22                                         status=400)

```

(c) Membuat data benih (HTTP Method - POST)

Fungsi ini digunakan untuk membuat data benih di database dengan method POST.

```

1      class SeedInventoriesApi(Resource):
2          @jwt_required()
3          def post(self):
4              try:
5                  current_user = get_jwt_identity()
6                  farm = str(current_user['farm_id'])
7                  body = {

```

```
8         "farm_id": farm,
9         "fish_seed_category": request.form.get(
10            "fish_seed_category", None),
11        "fish_type": request.form.get('fish_type', None),
12        "brand_name": request.form.get('brand_name', None),
13        "amount": request.form.get('amount', None),
14        "weight": request.form.get('weight', None),
15        "width": request.form.get('width', None),
16        "price": request.form.get('price', None),
17        "total_price": request.form.get('total_price', None),
18        "image": request.form.get('image', None)
19    }
20
21    inventory = SeedInventory(**body).save()
22
23    id = inventory.id
24
25    res = {"message": "success add seed to inventory", "id": id}
26    , "data": body}
27
28    response = json.dumps(res, default=str)
29
30    return Response(response, mimetype="application/json",
31 status=200)
32
33    except Exception as e:
34
35        response = {"message": str(e)}
36
37        response = json.dumps(response, default=str)
38
39        return Response(response, mimetype="application/json",
40 status=400)
```

Di fungsi ini, terdapat body yang berisi parameter form data yang akan dikirim dari frontend dan sesuai dengan model yang ada pada inventaris benih.

- (d) Memperbarui data benih secara spesifik berdasarkan ID benih (HTTP Method - PUT)

Fungsi ini digunakan untuk memperbarui data benih berdasarkan ID benih di database dengan method PUT.

```
1      class SeedInventoryApi(Resource):
2          def put(self, id):
3              try:
4                  body = {
5                      "id_int": int(id),
6                      "fish_seed_category": request.form.get('
fish_seed_category', None),
```

```

7      "fish_type": request.form.get('fish_type', None),
8      "brand_name": request.form.get('brand_name', None),
9      "amount": request.form.get("amount", None),
10     "weight": request.form.get("weight", None),
11     "width": request.form.get('width', None),
12     "price": request.form.get('price', None),
13     "total_price": request.form.get('total_price', None),
14     "image": request.form.get('image', None)
15   }
16   inventory = SeedInventory.objects.get(id_int = int(id)).
17   update(**body)
18   response = {"message": "success update seed inventory", "
19   data": body}
20   response = json.dumps(response, default=str)
21   return Response(response, mimetype="application/json",
22   status=200)
23 except Exception as e:
24   response = {"message": str(e)}
25   response = json.dumps(response, default=str)
26   return Response(response, mimetype="application/json",
27   status=400)
28

```

Fungsi ini kurang lebih sama seperti method POST karena memerlukan body yang sejenis, body ini nantinya yang akan dihubungkan ke database inventaris benih.

- (e) Menghapus data benih secara spesifik berdasarkan ID benih (HTTP Method - DELETE)

Fungsi ini digunakan untuk menghapus data benih berdasarkan ID benih di database dengan method DELETE.

```

1   class SeedInventoryApi(Resource):
2       def delete(self, id):
3           try:
4               inventory = SeedInventory.objects.get(id_int = int(id)).
5               delete()
6               response = {"message": "success delete seed inventory"}
7               response = json.dumps(response, default=str)
8               return Response(response, mimetype="application/json",
9               status=200)
10          except Exception as e:
11

```

```

9      response = {"message": str(e)}
10     response = json.dumps(response, default=str)
11     return Response(response, mimetype="application/json",
12                      status=400)

```

Setelah class selesai dibuat, diperlukan route yang digunakan untuk menyambungkan ke class tersebut. Route ini dapat dilihat pada sample route yang sudah dibuat sebelumnya. Berikut cara penyambungan route terhadap class pada Flask :

```

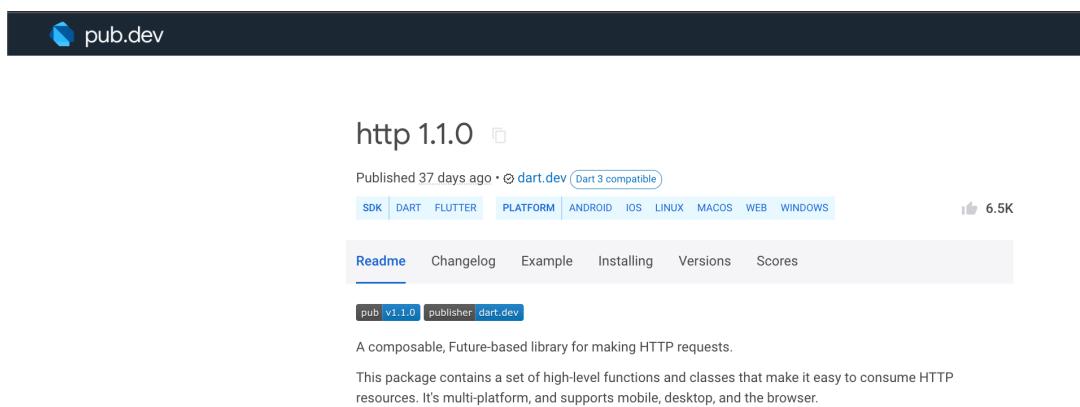
1 # seed inventory
2 api.add_resource(SeedInventoriesApi, '/api/inventory/seed')
3 api.add_resource(SeedInventoryApi, '/api/inventory/seed/<id>')
4

```

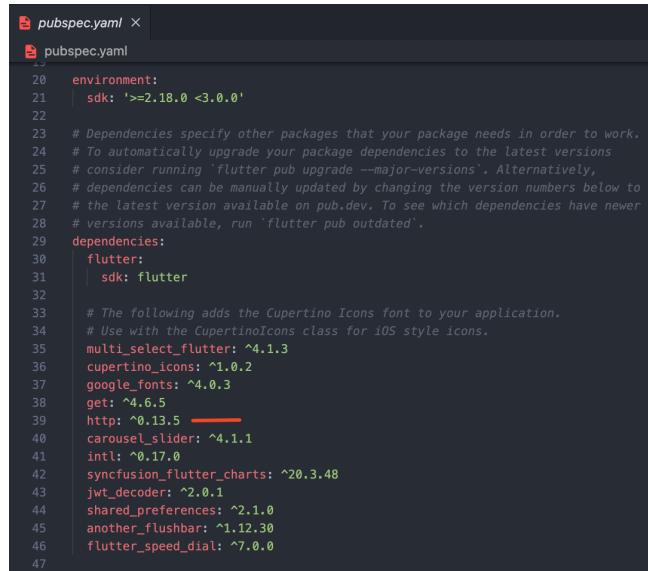
Route tersebut nantinya yang akan digunakan Flutter untuk berkomunikasi pada backend sesuai dengan HTTP Request yang ada pada class tersebut.

2. Membuat halaman serta Integrasi RESTful API benih dengan Flutter

Pada Flutter, untuk berkomunikasi pada backend diperlukan package HTTP yang perlu ditambahkan terlebih dahulu.



Gambar 4.26: Package HTTP untuk Flutter



```

20   environment:
21     sdk: '>=2.18.0 <3.0.0'
22
23   # Dependencies specify other packages that your package needs in order to work.
24   # To automatically upgrade your package dependencies to the latest versions
25   # consider running `flutter pub upgrade --major-versions` . Alternatively,
26   # dependencies can be manually updated by changing the version numbers below to
27   # the latest version available on pub.dev. To see which dependencies have newer
28   # versions available, run `flutter pub outdated` .
29   dependencies:
30     flutter:
31       sdk: flutter
32
33   # The following adds the Cupertino Icons font to your application.
34   # Use with the CupertinoIcons class for iOS style icons.
35   multi_select_flutter: ^4.1.3
36   cupertino_icons: ^1.0.2
37   google_fonts: ^4.0.3
38   get: ^4.6.5
39   http: ^0.13.5 _____
40   carouse_slider: ^4.1.1
41   intl: ^0.17.0
42   syncfusion_flutter_charts: ^20.3.48
43   jwt_decoder: ^2.0.1
44   shared_preferences: ^2.1.0
45   another_flushbar: ^1.12.30
46   flutter_speed_dial: ^7.0.0
47

```

Gambar 4.27: Penambahan package HTTP pada Flutter

Setelah package HTTP ter-install, dibuatlah controller yang berisi fungsi-fungsi HTTP request yang sesuai dengan class route pada backend.

Untuk keseluruhan fungsi HTTP request pada Flutter, dapat dilihat pada baris kode dibawah ini.

(a) Mengambil semua data benih (HTTP Method - GET)

Fungsi ini digunakan untuk mengambil semua data benih, baik itu jenis benih ataupun pembesaran dari segala jenis ikan.

```

1      Future getAllSeedData(String type) async {
2
3         SharedPreferences prefs = await SharedPreferences.getInstance()
4         ;
5
6         String token = prefs.getString('token').toString();
7
8         seedList.value.data!.clear();
9         nameHistoryList.clear();
10        nameHistoryList.add('Semua');
11        listMas.clear();
12        listNilaHitam.clear();
13        listNilaMerah.clear();
14        listPatin.clear();
15        listLele.clear();

```

```
13     resetVariables();
14     isLoadingPage.value = true;
15
16     var headers = {'Authorization': 'Bearer $token'};
17     final response = await http.get(
18       Uri.parse('${Urls.invSeed}?type=$type'),
19       headers: headers,
20     );
21
22     try {
23       if (response.statusCode == 200) {
24         InventarisBenihModel res =
25           InventarisBenihModel.fromJson(jsonDecode(response.body));
26
27         seedList.value = res;
28
29         for (var i in seedList.value.data!) {
30           nameHistoryList.add(i.brandName.toString());
31         }
32
33         for (var i in seedList.value.data!) {
34           if (i.fishType == 'Lele') {
35             listLele.add({
36               'id': i.idInt,
37               'seed_id': i.sId,
38               'fishName': i.brandName,
39             });
40
41             selectedLele.value = listLele[0];
42           }
43           if (i.fishType == 'Nila Hitam') {
44             listNilaHitam.add({
45               'id': i.idInt,
46               'seed_id': i.sId,
47               'fishName': i.brandName,
48             });
49
50             selectedNilaHitam.value = listNilaHitam[0];
51           }
52           if (i.fishType == 'Nila Merah') {
53             listNilaMerah.add({
54               'id': i.idInt,
55               'seed_id': i.sId,
56               'fishName': i.brandName,
```

```

57     });
58
59     selectedNilaMerah.value = listNilaMerah[0];
60 }
61 if (i.fishType == 'Patin') {
62     listPatin.add({
63     'id': i.idInt,
64     'seed_id': i.sId,
65     'fishName': i.brandName,
66 });
67
68     selectedPatin.value = listPatin[0];
69 }
70 if (i.fishType == 'Mas') {
71     listMas.add({
72     'id': i.idInt,
73     'seed_id': i.sId,
74     'fishName': i.brandName,
75 });
76
77     selectedMas.value = listMas[0];
78 }
79 }
80
81     selectedNameHistory.value = nameHistoryList[0];
82 }
83 // inspect(listLele);
84 } catch (e) {
85     throw Exception(e);
86 }
87 isLoadingPage.value = false;
88 }
89

```

Parameter yang digunakan fungsi ini adalah type, type disini mewakili kategori benih yang ingin diambil. Pada proses pengambilan data benih, benih disimpan didalam variabel masing-masing yang diinisialsasikan sebagai list. Hasil dari request tersebut ditampung dalam variabel res yang merepresentasikan model inventaris benih.

(b) Mengambil data benih secara spesifik berdasarkan ID benih (HTTP

Method - GET)

Fungsi ini digunakan untuk mengambil data benih secara spesifik berdasarkan ID benih nya.

```

1     Future getSeedDataByID(int id, Function() doAfter) async {
2         isLoadingDetail.value = true;
3         final response = await http.get(Uri.parse('${Urls.invSeed}/$id'
4             ));
5
6         try {
7             if (response.statusCode == 200) {
8                 DetailInventarisBenihModel res =
9                     DetailInventarisBenihModel.fromJson(jsonDecode(response.
10                     body));
11
12                     seedCategory.value = res.data!.fishSeedCategory.toString();
13                     fishCategory.value = res.data!.fishType.toString();
14                     sortSize.value = res.data!.width.toString();
15                     fishName.value = seedCategory.value == 'Benih'
16                         ? '${fishCategory.value}${sortSize.value.split(' ')[0].
17                         replaceAll('-', '')}'
18                         : '${fishCategory.value}${fishWeight.text.split(' ')[0]}';
19                     fishAmount.text = res.data!.amount.toString();
20                     fishWeight.text = res.data!.weight!.toStringAsFixed(2);
21                     fishPrice.text = res.data!.price.toString();
22                     fishPriceTotal.text = res.data!.totalPrice.toString();
23                     fishImage.value = res.data!.image.toString();
24             }
25             doAfter();
26         } catch (e) {
27             throw Exception(e);
28         }
29         isLoadingDetail.value = false;
30     }
31 
```

Paramter fungsi ini tentunya adalah ID benih yang digunakan pada endpoint. Hasil dari request tersebut ditampung dalam variabel res yang merepresentasikan model detail inventaris benih dan masing-masing response diwakili satu per satu oleh variabel yang sesuai.

- (c) Membuat data benih (HTTP Method - POST)

Fungsi ini digunakan untuk membuat data benih yang akan dikirimkan ke database.

```

1     Future postSeedData(Function() doAfter) async {
2         var map = <String, dynamic>{};
3
4         SharedPreferences prefs = await SharedPreferences.getInstance()
5         ;
6         String token = prefs.getString('token').toString();
7         var headers = {'Authorization': 'Bearer $token'};
8
9         map['fish_seed_category'] = seedCategory.value;
10        map['fish_type'] = fishCategory.value;
11        map['brand_name'] = seedCategory.value == 'Benih'
12            ? '${fishCategory.value.replaceAll(' ', '')}${sortSize.value.
13            split(' ')[0]}'
14            : '${fishCategory.value.replaceAll(' ', '')}${fishWeight.text
15            .replaceAll(',', '.').split('.')[0]}';
16        map['amount'] = fishAmount.text == '' ? '0' : fishAmount.text;
17        map['weight'] =
18            fishWeight.text == '' ? '0' : fishWeight.text.replaceAll(',', '
19            .');
20        map['width'] = seedCategory.value == 'Benih' ? sortSize.value :
21            "";
22
23        map['price'] = fishPrice.text == '' ? '0' : fishPrice.text;
24        map['total_price'] = fishPriceTotal.text == '' ? '0' :
25            fishPriceTotal.text;
26        map['image'] = fishImage.value;
27
28        isLoadingPost.value = true;
29
30        inspect(map);
31
32        try {
33            await http.post(
34                Uri.parse(Urls.invSeed),
35                body: map,
36                headers: headers,
37            );
38            doAfter();
39        } catch (e) {
40            throw Exception(e);
41        }
42        isLoadingPost.value = false;

```

```

36 }
37

```

Untuk mengirim data body ke database, di Flutter menggunakan map untuk merepresentasikan value yang akan diterima pada backend nantinya. Jika value body tidak sesuai dengan yang ada di model backend, maka backend akan merespon error dan data tidak dapat masuk ke database.

(d) Memperbarui data benih secara spesifik (HTTP Method - PUT)

Fungsi ini digunakan untuk memperbarui data benih secara spesifik berdasarkan ID benih.

```

1 Future updateSeedData(int id, Function() doAfter) async {
2     var map = <String, dynamic>{};
3
4     map['fish_seed_category'] = seedCategory.value;
5     map['fish_type'] = fishCategory.value;
6     map['brand_name'] = seedCategory.value == 'Benih'
7         ? '${fishCategory.value.replaceAll(' ', '')}${sortSize.value.
split(' ')[0]}'
8         : '${fishCategory.value.replaceAll(' ', '')}${fishWeight.text
.replaceAll(',', '.').split('.')[0]}';
9     map['amount'] = fishAmount.text == '' ? '0' : fishAmount.text;
10    map['weight'] =
11        fishWeight.text == '' ? '0' : fishWeight.text.replaceAll(',', '.');
12    map['width'] = seedCategory.value == 'Benih' ? sortSize.value :
"";
13    map['price'] = fishPrice.text == '' ? '0' : fishPrice.text;
14    map['total_price'] = fishPriceTotal.text == '' ? '0' :
fishPriceTotal.text;
15    map['image'] = fishImage.value;
16
17    isLoadingPost.value = true;
18
19    try {
20        inspect(map);
21        await http.put(
22            Uri.parse('${Urls.invSeed}/$id'),
23            body: map,

```

```

24     );
25     doAfter();
26   } catch (e) {
27     throw Exception(e);
28   }
29   isLoadingPost.value = false;
30 }
31

```

Fungsi ini kurang lebih sama seperti fungsi POST request, bedanya hanya method nya saja. Untuk memperbarui data, digunakan method PUT dan value yang diterima merupakan body yang sudah di map.

(e) Menghapus data benih secara spesifik (HTTP Method - DELETE)

Fungsi ini digunakan untuk menghapus data benih secara spesifik sesuai dengan ID benih.

```

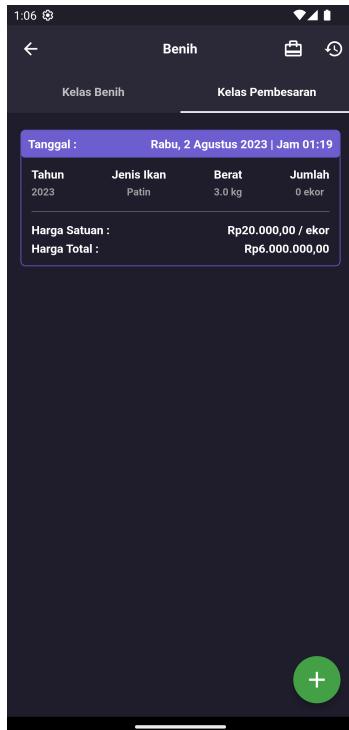
1   Future deleteSeedData(int id, Function() doAfter) async {
2     isLoadingDelete.value = true;
3     try {
4       await http.delete(
5         Uri.parse(
6           '${Urls.invSeed}/$id',
7         ),
8         headers: {
9           'Content-Type': 'application/json; charset=UTF-8',
10          },
11        );
12       doAfter();
13     } catch (e) {
14       throw Exception(e);
15     }
16     isLoadingDelete.value = false;
17   }
18

```

Parameter yang digunakan pada fungsi ini adalah ID benih dan method HTTP yang digunakan adalah DELETE.

Setelah fungsi diatas sudah berjalan, diperlukan layout pada Flutter yang nantinya akan ditampilkan kepada user sehingga user dapat menggunakan

aplikasi untuk fitur inventaris benih. Layout ini nantinya akan diintegrasikan dengan fungsi-fungsi HTTP request pada controller inventaris benih. Berikut layout atau tampilan dari inventaris benih. Untuk code dapat dilihat pada halaman Lampiran.



Gambar 4.28: Halaman Inventaris Benih



Gambar 4.29: Halaman Input Inventaris Benih



Gambar 4.30: Halaman Detail Inventaris Benih

(a) Halaman Inventaris Benih

Pada halaman ini, terdapat list dari benih yang dibagi menjadi dua kelas yaitu kelas benih dan kelas pembesaran. Pada bagian header layar, terdapat tombol tas untuk menuju ke menu inventaris dan tombol riwayat untuk masuk ke halaman riwayat penggunaan benih.

Di pojok kanan bawah, terdapat tombol (+) yang berfungsi untuk menavigasikan ke halaman input inventaris benih.

(b) Halaman Input Inventaris Benih

Halaman ini berisi form-form input yang diperlukan untuk pendataan

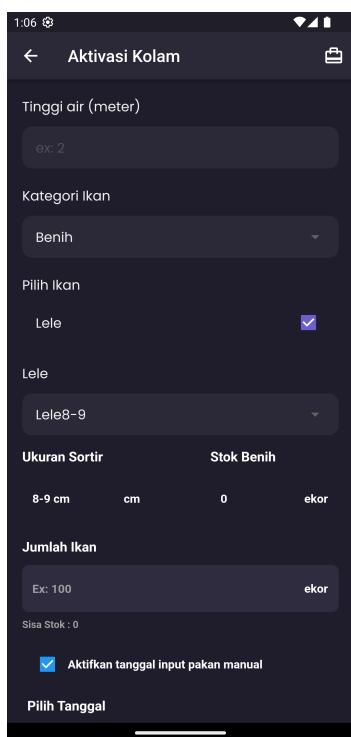
benih.

(c) Halaman Detail Inventaris Benih

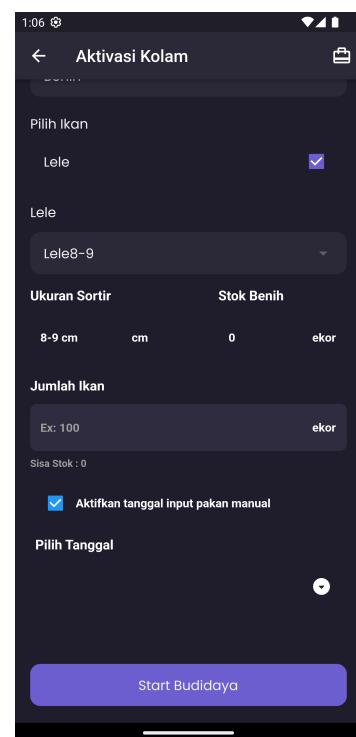
Halaman ini serupa dengan halaman input inventaris benih, yang membedakan adalah

Dari tampilan tersebut, bagian yang membedakan dengan halaman input inventaris benih adalah form nya tidak langsung dapat di edit dan terdapat dua tombol tambahan yaitu tombol Edit untuk memperbarui data dan tombol Hapus untuk menghapus data.

Untuk inventaris benih digunakan pada saat aktivasi kolam dilakukan. Berikut merupakan tampilan dari aktivasi kolam yang sudah diperbarui dengan data inventaris benih.



Gambar 4.31: Halaman Aktivasi Kolam



Gambar 4.32: Halaman Aktivasi Kolam

Pada halaman tersebut, dapat dilihat benih yang tersedia pada inventaris beserta dengan informasi-informasi mengenai benih tersebut.

3. Membuat model riwayat pemakaian benih

Dalam penggunaan inventaris benih, diperlukan fitur riwayat penggunaan benih untuk melihat pengeluaran benih ikan tiap musim budidaya. Untuk membuat fitur ini, hal yang pertama dilakukan adalah membuat model dari riwayat pemakaian benih. Model dapat dilihat sebagai berikut.

```

1   class SeedUsed(db.Document):
2       fish_seed_id = db.ReferenceField(SeedInventory, required=True)
3       farm_id = db.ReferenceField(Farm, required=True)
4       original_amount = db.IntField(required=True)
5       usage = db.IntField(required=True)
6       pond = db.StringField(required=True)
7       created_at = db.DateTimeField(default=datetime.datetime.now)
8       updated_at = db.DateTimeField(default=datetime.datetime.now)
9

```

4. Design route dan penerapan dengan Flutter untuk riwayat pemakaian benih (dalam bentuk RESTful API)

Setelah model selesai, dibuat route yang kemudian diintegrasikan kepada class riwayat benih pada backend.

Group	Endpoint	HTTP Status	Operation	Purpose
History Inventaris Benih	/history/inventory/seed?start_date=&end_date=&pond_name=&name=	GET	READ	mengambil data riwayat penggunaan benih
	/history/inventory/seed	POST	CREATE	menginput data penggunaan benih

Gambar 4.33: Sample Route Riwayat Benih

```

1   api.add_resource(SeedHistoryApi, '/api/history/inventory/seed')
2

```

Beberapa fungsi dari class riwayat benih ini dapat dilihat pada kode dibawah ini.

(a) Mengambil data riwayat pemakaian benih (HTTP Method - GET)

```

1   class SeedHistoryApi(Resource):
2       @jwt_required()
3
4       def get(self):

```



```
41         '$match': {
42             'brand_name': {
43                 '$regex': name,
44                 '$options': 'i'
45             }
46         }
47     },
48     {"$project": {
49         "_id": 1,
50         "fish_seed_category": 1,
51         "fish_type": 1,
52         "brand_name": 1,
53         "price": 1,
54         "created_at": 1,
55     } }
56 ],
57     'as': 'seed'
58 } },
59 {"$addFields": {
60     "seed": {"$first": "$seed"},
61 } },
62 ]
63
64 testing = SeedUsed.objects.aggregate(pipeline)
65 temp = list(testing)
66 result = []
67
68 for i in temp:
69     if 'seed' in i:
70         result.append(i)
71
72
73 response = json.dumps({
74     'status': 'success',
75     'data': result,
76 }, default=str)
77 return Response(response, mimetype="application/json",
status=200)
78 except Exception as e:
79     response = {"message": e}
80 response = json.dumps(response, default=str)
81 return Response(response, mimetype="application/json",
status=400)
```

Pada fungsi tersebut, terdapat empat query params yang berguna untuk memfilter data yaitu start_date yang bernilai tanggal awal, end_date yang bernilai tanggal akhir, name yang bernilai nama benih, dan pond_name yang bernilai nama kolam.

Untuk response data riwayat benih, disini dilakukan \$lookup pada pipeline untuk mengambil data pada database inventaris benih untuk menampilkan detail data benih.

(b) Membuat data riwayat pemakaian benih (HTTP Method - POST)

```
1 class SeedHistoryApi(Resource):
2     @jwt_required()
3     def post(self):
4         try:
5             current_user = get_jwt_identity()
6             farm = str(current_user['farm_id'])
7
8             req_pond = request.form.get('pond', None)
9             req_seed_id = request.form.get('fish_seed_id', None)
10            req_usage = request.form.get('usage', None)
11
12            history_by_pond = SeedUsed.objects(pond=req_pond,
13                                              fish_seed_id=req_seed_id).first()
14
15            print(history_by_pond)
16
17            theDate = request.form.get('created_at', None)
18
19            body = {
20                "farm_id": farm,
21                "fish_seed_id": request.form.get('fish_seed_id', None),
22                "original_amount": request.form.get('original_amount',
23                                              None),
24                "usage": request.form.get('usage', None),
25                "pond": request.form.get('pond', None),
26            }
27
28            if theDate != '':
29                body['created_at'] = datetime.datetime.strptime(theDate,
30                                                               "%Y-%m-%dT%H:%M:%S.%f %z")
31
32        except Exception as e:
33            return {'error': str(e)}, 500
34
35    def get(self, farm_id):
36        try:
37            farm = Farm.objects(id=farm_id).first()
38
39            if farm:
40                return {'farm': farm.to_json()}
41            else:
42                return {'error': 'Farm not found'}, 404
43
44        except Exception as e:
45            return {'error': str(e)}, 500
46
47    def put(self, farm_id):
48        try:
49            farm = Farm.objects(id=farm_id).first()
50
51            if farm:
52                farm.name = request.json['name']
53                farm.save()
54
55                return {'farm': farm.to_json()}, 200
56            else:
57                return {'error': 'Farm not found'}, 404
58
59        except Exception as e:
60            return {'error': str(e)}, 500
61
62    def delete(self, farm_id):
63        try:
64            farm = Farm.objects(id=farm_id).first()
65
66            if farm:
67                farm.delete()
68
69                return {'message': 'Farm deleted successfully'}, 200
70            else:
71                return {'error': 'Farm not found'}, 404
72
73        except Exception as e:
74            return {'error': str(e)}, 500
```

```
28
29         except Exception as e:
30             response = {"message": str(e)}
31             response = json.dumps(response, default=str)
32             return Response(response, mimetype="application/json",
33                             status=400)
```

Setelah fungsi selesai, dibuat controller riwayat benih pada Flutter yang berisi fungsi yang sesuai pada class riwayat benih.

(a) Mengambil data riwayat pemakaian benih (HTTP Method - GET)

```

27     }
28
29     doAfter();
30   }
31 } catch (e) {
32   throw Exception(e);
33 }
34 isLoadingHistory.value = false;
35 }
36

```

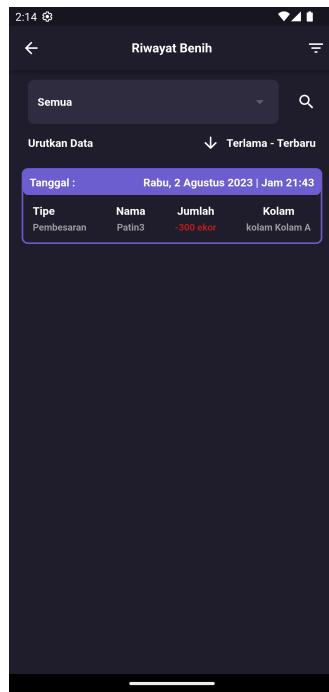
(b) Membuat data riwayat pemakaian benih (HTTP Method - POST)

```

1   Future postHistorySeedData(
2     String pondName, List fish, String usedDate, Function() doAfter
3   ) async {
4     var map = <String, dynamic>{};
5
6     map['pond'] = pondName;
7
8     SharedPreferences prefs = await SharedPreferences.getInstance()
9     ;
10
11    String token = prefs.getString('token').toString();
12    var headers = {'Authorization': 'Bearer $token'};
13
14    for (var i = 0; i < fish.length; i++) {
15      map['fish_seed_id'] = fish[i]['seed_id'];
16      map['original_amount'] = fish[i]['original_value'];
17      map['usage'] = fish[i]['amount'];
18      map['created_at'] = usedDate;
19
20      try {
21        await http.post(
22          Uri.parse(Urls.seedSch),
23          body: map,
24          headers: headers,
25        );
26        doAfter();
27      } catch (e) {
28        throw Exception(e);
29      }
30    }
31
32  }
33
34
35
36

```

Setelah controller sudah siap digunakan, untuk tampilan dari riwayat pemakaian benih dapat dilihat pada layout dan code berikut.



Gambar 4.34: Halaman Penggunaan Benih

Pada halaman tersebut, terdapat list dari penggunaan benih. Rincian data yang ditampilkan adalah tipe, nama, jumlah, dan tempat kolam benih diletakkan. Terdapat juga beberapa layout yang digunakan untuk memfilter data.

5. Sprint 3 Review

Hasil review pada Sprint 3 ini adalah review dan testing oleh penulis selaku developer dengan Scrum Master. Setelah dilakukan testing, Scrum Master menyimpulkan bahwa penerapan fitur inventaris benih dan riwayat benih telah berjalan dengan baik.

4. Sprint 4

Sprint 4 dilaksanakan pada tanggal 21 Juni 2023 - 12 Juli 2023. Detail dari Sprint 4 ini adalah mengerjakan tugas yang ada pada Sprint 4 Backlog di tabel berikut.

Tabel 4.4: Sprint 4 Backlog

No	Stories	Task	Status
1	Fitur pencatatan inventaris	- Design route dan penerapan pada Flutter untuk inventaris pakan (dalam bentuk RESTful API)	Selesai
		- Design route dan penerapan pada Flutter untuk inventaris suplemen (dalam bentuk RESTful API)	Selesai
		- Design route dan penerapan pada Flutter untuk inventaris listrik (dalam bentuk RESTful API)	Selesai
		- Design route dan penerapan pada Flutter untuk inventaris aset (dalam bentuk RESTful API)	Selesai
		- Design route dan penerapan pada Flutter untuk riwayat pemakaian pakan (dalam bentuk RESTful API)	Selesai
		- Design route dan penerapan pada Flutter untuk riwayat pemakaian suplemen (dalam bentuk RESTful API)	Selesai
2	Fitur Pemberian pakan yang terkoneksi dengan inventaris	- Integrasi data inventaris pakan pada halaman entry pakan	Selesai

Dalam proses Sprint ini, beberapa tahapan yang dilakukan adalah sebagai berikut :

1. Design route dan penerapan pada Flutter untuk inventaris pakan (dalam bentuk RESTful API)

Untuk melakukan tugas ini, langkah-langkahnya sama seperti yang ada pada Sprint 3. Langkah-langkah yang dilakukan adalah sebagai berikut:

(a) Design Sample Route

Berikut merupakan sample route yang sudah dibuat untuk inventaris pakan.

Group	Endpoint	HTTP Status	Operation	Purpose
Inventaris Pakan	/inventory/feed?type=	GET	READ	mengambil semua data inventaris pakan dengan filter type
	/inventory/feed/\${id}	GET	READ	mengambil data spesifik dari inventaris pakan dengan id
	/inventory/feed	POST	CREATE	menginput data kedalam inventaris pakan
	/inventory/feed/\${id}	PUT	UPDATE	memperbarui spesifik data pada inventaris pakan dengan id
	/inventory/feed/\${id}	DELETE	DELETE	menghapus spesifik data pada inventaris pakan dengan id

Gambar 4.35: Sample Route Inventaris Pakan

(b) Model Backend

Berdasarkan sample route tersebut, dapat dibuat model dan class HTTP method pada backend. Berikut model serta class untuk inventaris pakan :

```

1      class FeedInventory(db.Document):
2          feed_name_id = db.ReferenceField(FeedName, required=True)
3          farm_id = db.ReferenceField(Farm, required=True)
4          id_int = db.SequenceField(required=True)
5          feed_category = db.StringField(required=True)
6          brand_name = db.StringField(required=True)
7          price = db.IntField(required=True)
8          amount = db.FloatField(required=True)
9          created_at = db.DateTimeField(default=datetime.datetime.now)
10         updated_at = db.DateTimeField(default=datetime.datetime.now)
11

```

(c) Fungsi-fungsi HTTP Method

- Mengambil semua data inventaris pakan (HTTP Method - GET)

```
1 class FeedInventoriesApi(Resource):
2     @jwt_required()
3
4     def get(self):
5         try:
6             current_user = get_jwt_identity()
7             farm = str(current_user['farm_id'])
8             farm_id = ObjectId(farm)
9
10            type = request.args.get('type') if request.args.get(
11 'type') else ""
12
13            pipeline = [
14                {"$sort": {"id_int": 1}},
15                {
16                    '$match': {
17                        "farm_id": farm_id,
18                        'feed_category': {
19                            '$regex': type,
20                            '$options': 'i'
21                        }
22                    }
23                },
24                {'$lookup': {
25                    'from': 'feed_name',
26                    'let': {"feednameid": "$feed_name_id"},
27                    'pipeline': [
28                        {'$match': {'$expr': {'$eq': ['$id', '$$feednameid']}}, {
29                            '$project': {
30                                "_id": 1,
31                                "id_int": 1,
32                                "type": 1,
33                                "name": 1,
34                                "description": 1,
35                                "producer": 1,
36                                "protein": 1,
37                                "carbohydrate": 1,
38                                "min_expired_period": 1,
39                                "max_expired_period": 1,
40                                "image": 1,
41                                "created_at": 1,
42                            }
43                        }
44                    ]
45                }
46            }
47        }
48    }
```

```

42         ],
43         'as': 'feed'
44     } },
45     {"$addFields": {
46         "feed": {"$first": "$feed"},
47     } },
48 ]
49
50     testing = FeedInventory.objects.aggregate(pipeline)
51     temp = list(testing)
52     response = json.dumps({
53         'status': 'success',
54         'data': temp,
55     }, default=str)
56     return Response(response, mimetype="application/json",
57     ", status=200)
58 except Exception as e:
59     response = {"message": e}
60     response = json.dumps(response, default=str)
61     return Response(response, mimetype="application/json",
62     ", status=400)
63

```

- Mengambil spesifik data inventaris pakan (HTTP Method - GET)

```

1     class FeedInventoryApi(Resource):
2         def get(self, id):
3             try:
4                 pipeline = [
5                     {"$match": {"id_int": int(id)}},
6                     {'$lookup': {
7                         'from': 'feed_name',
8                         'let': {"feednameid": "$feed_name_id"},
9                         'pipeline': [
10                             {"$match": {"$expr": {"$eq": ["$_id', '$
11 $feednameid']}}, },
12                             {"$project": {
13                                 "_id": 1,
14                                 "id_int": 1,
15                                 "type": 1,
16                                 "name": 1,
17                                 "description": 1,
18                                 "producer": 1,
19                                 "protein": 1,
20                                 "calories": 1
21                             }
22                         }
23                     }
24                 ]
25                 testing = FeedInventory.objects.aggregate(pipeline)
26                 temp = list(testing)
27                 response = json.dumps({
28                     'status': 'success',
29                     'data': temp,
30                 }, default=str)
31                 return Response(response, mimetype="application/json",
32                 ", status=200)
33             except Exception as e:
34                 response = {"message": e}
35                 response = json.dumps(response, default=str)
36                 return Response(response, mimetype="application/json",
37                 ", status=400)
38

```

```
19             "carbohydrate": 1,
20             "min_expired_period": 1,
21             "max_expired_period": 1,
22             "image": 1,
23             "created_at": 1,
24         } }
25     ],
26     'as': 'feed'
27 },
28 {"$addFields": {
29     "feed": {"$first": "$feed"},
30 },
31 ]
32
33 testing = FeedInventory.objects.aggregate(pipeline)
34 temp = list(testing)
35 if len(temp) == 0:
36     res = {"message": 'no data found'}
37     response = json.dumps(res, default=str)
38     return Response(response, mimetype="application/
39 json", status=200)
40     response = json.dumps({
41         'status': 'success',
42         'data': temp[0],
43     }, default=str)
44     return Response(response, mimetype="application/json
45 ", status=200)
46     except Exception as e:
47         response = {"message": e}
48         response = json.dumps(response, default=str)
49         return Response(response, mimetype="application/json
50 ", status=400)
```

- Menambahkan data inventaris pakan (HTTP Method - POST)

```
1     class FeedInventoriesApi(Resource):
2
3         @jwt_required()
4
5         def post(self):
6
7             try:
8
9                 current_user = get_jwt_identity()
10
11                 farm = str(current_user['farm_id'])
12
13                 body = {
14
15                     "feed_name_id": request.form.get('feed_name_id',
```

```
None),
9         "feed_category": request.form.get('feed_category'),
10        None),
11
12        "brand_name": request.form.get('brand_name', None)
13
14        ,
15        "price": request.form.get('price', None),
16        "amount": request.form.get('amount', None),
17        }
18
19        inventory = FeedInventory(**body).save()
20
21        id = inventory.id
22
23        res = {"message": "success add feed to inventory",
24               "id": id, "data": body}
25
26        response = json.dumps(res, default=str)
27
28        return Response(response, mimetype="application/json"
29                      ),
30        status=200)
31
32        except Exception as e:
33
34            response = {"message": str(e)}
35
36            response = json.dumps(response, default=str)
37
38            return Response(response, mimetype="application/json"
39                          ),
40            status=400)
```

- Memperbarui spesifik data inventaris pakan (HTTP Method - PUT)

```
1 class FeedInventoryApi(Resource):
2     def put(self, id):
3         try:
4             body = {
5                 "id_int": int(id),
6                 "feed_category": request.form.get('feed_category',
7                     None),
8                 "brand_name": request.form.get('brand_name', None)
9             ,
10                "price": request.form.get('price', None),
11                "amount": request.form.get('amount', None),
12            }
13
14            feed_name_id = request.form.get('feed_name_id', None)
15        )
16
17            feed_name = FeedName.objects.get(id=feed_name_id)
18            body["feed_name_id"] = feed_name.id
19
20
21            inventory = FeedInventory.objects.get(id_int = int(
22                id)).update(**body)
```

```

17         response = {"message": "success update feed
inventory", "data": body}
18             response = json.dumps(response, default=str)
19             return Response(response, mimetype="application/json
", status=200)
20     except Exception as e:
21         response = {"message": str(e)}
22         response = json.dumps(response, default=str)
23         return Response(response, mimetype="application/json
", status=400)
24

```

- Menghapus spesifik data inventaris pakan (HTTP Method - DELETE)

```

1     class FeedInventoryApi(Resource):
2         def delete(self, id):
3             try:
4                 inventory = FeedInventory.objects.get(id_int = int(
id)).delete()
5                 response = {"message": "success delete feed
inventory"}
6                 response = json.dumps(response, default=str)
7                 return Response(response, mimetype="application/json
", status=200)
8             except Exception as e:
9                 response = {"message": str(e)}
10                response = json.dumps(response, default=str)
11                return Response(response, mimetype="application/json
", status=400)
12

```

(d) Controller HTTP pada Flutter

- Mengambil semua data inventaris pakan (HTTP Method - GET)

```

1     Future getAllData(String type, Function() doAfter) async {
2         feedList.value.data!.clear();
3         selectedFeedList.clear();
4         isLoadingPage.value = true;
5
6         SharedPreferences prefs = await SharedPreferences.
getInstance();
7         String token = prefs.getString('token').toString();

```

```

8         var headers = {'Authorization': 'Bearer $token'};
9
10        final response = await http.get(
11          Uri.parse('${Urls.invFeed}?type=$type'),
12          headers: headers,
13        );
14
15        try {
16          if (response.statusCode == 200) {
17            InventarisPakanModel res =
18              InventarisPakanModel.fromJson(jsonDecode(response.
19                body));
20
21            feedList.value = res;
22
23            for (var i in feedList.value.data!) {
24              selectedFeedList.add({
25                'id': i.idInt,
26                'feed_id': i.sId,
27                'feed_name': i.brandName,
28              });
29
30            selectedFeedName.value = selectedFeedList[0];
31
32            inspect(feedList.value.data);
33
34            doAfter();
35          }
36        } catch (e) {
37          throw Exception(e);
38        }
39        isLoadingPage.value = false;
40      }
41

```

- Mengambil spesifik data inventaris pakan (HTTP Method - GET)

```

1   Future getDataByID(int id, Function() doAfter) async {
2     // resetVariables();
3     isLoadingDetail.value = true;
4     isLoadingFeedDetail.value = true;
5
6     final response = await http.get(Uri.parse('${Urls.

```

```

    invFeed} / $id' ));

7
8     try {
9         if (response.statusCode == 200) {
10            DetailInventarisPakanModel res =
11                DetailInventarisPakanModel.fromJson(jsonDecode(
12                    response.body));
13
14            feedCategory.value = res.data!.feedCategory!;
15
16            for (var i in listPakanName) {
17                if (i['feed_name_id'] == res.data!.feedNameId) {
18                    selectedPakan.value = i;
19                }
20            }
21
22            // desc.text = res.data!.description.toString();
23            price.text = res.data!.price.toString();
24            amount.text = res.data!.amount!.toStringAsFixed(2);
25            producer.text = res.data!.feed!.producer.toString();
26            protein.text = res.data!.feed!.protein.toString();
27            carbo.text = res.data!.feed!.carbohydrate.toString();
28            minExp.text = res.data!.feed!.minExpiredPeriod.
29            toString();
29            // maxExp.text = res.data!.maxExpiredPeriod.toString()
30            ;
31            // image.value = res.data!.image.toString();
32        }
33        doAfter();
34    } catch (e) {
35        throw Exception(e);
36    }
37    isLoadingDetail.value = false;
38    isLoadingFeedDetail.value = false;
38

```

- Menambahkan data inventaris pakan (HTTP Method - POST)

```

1     Future postData(Function() doAfter) async {
2         var map = <String, dynamic>{};
3
4         map['feed_name_id'] = selectedPakan.value['feed_name_id']
5     };

```

```

5      map['feed_category'] = feedCategory.value;
6      map['brand_name'] = selectedPakan.value['feed_name'];
7      map['price'] = price.text;
8      map['amount'] = amount.text.replaceAll(',', '.');
9
10     SharedPreferences prefs = await SharedPreferences.
11         getInstance();
12     String token = prefs.getString('token').toString();
13     var headers = {'Authorization': 'Bearer $token'};
14
15     isLoadingPost.value = true;
16
17     inspect(map);
18
19     try {
20         await http.post(
21             Uri.parseUrls.invFeed),
22             body: map,
23             headers: headers,
24         );
25         doAfter();
26     } catch (e) {
27         throw Exception(e);
28     }
29     isLoadingPost.value = false;
30 }
```

- Memperbarui spesifik data inventaris pakan (HTTP Method - PUT)

```

1     Future updateData(int id, Function() doAfter) async {
2         var map = <String, dynamic>{};
3
4         map['feed_name_id'] = selectedPakan.value['feed_name_id'];
5
6         map['feed_category'] = feedCategory.value;
7         map['brand_name'] = selectedPakan.value['feed_name'];
8         map['price'] = price.text;
9         map['amount'] = amount.text.replaceAll(',', '.');
10
11        isLoadingPost.value = true;
12
13        try {
14            inspect(map);
```

```

14         await http.put(
15             Uri.parse('${Urls.invFeed}/$id'),
16             body: map,
17         );
18         doAfter();
19     } catch (e) {
20         throw Exception(e);
21     }
22     isLoadingPost.value = false;
23 }
24

```

- Menghapus spesifik data inventaris pakan (HTTP Method - DELETE)

```

1     Future deleteData(int id, Function() doAfter) async {
2         isLoadingDelete.value = true;
3         try {
4             await http.delete(
5                 Uri.parse(
6                     '${Urls.invFeed}/$id',
7                 ),
8                 headers: {
9                     'Content-Type': 'application/json; charset=UTF-8',
10                },
11            );
12            doAfter();
13        } catch (e) {
14            throw Exception(e);
15        }
16        isLoadingDelete.value = false;
17    }
18

```

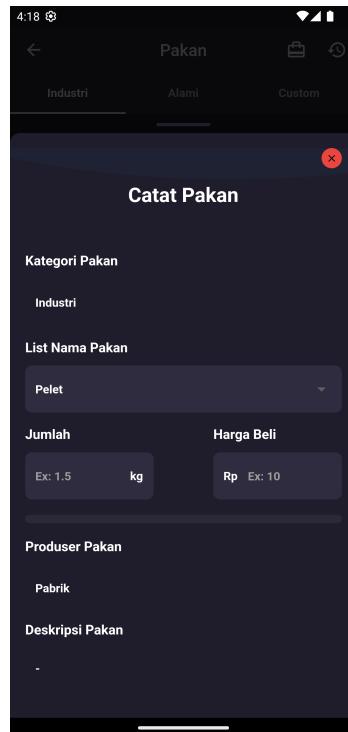
Untuk Method GET, diperlukan model Flutter yang merepresentasikan model yang ada pada backend.

(e) Tampilan pada Flutter

- Halaman Inventaris Pakan, Input Pakan, dan Detail Pakan



Gambar 4.36: Halaman Inventaris Pakan



Gambar 4.37: Halaman Input Inventaris Pakan



Gambar 4.38: Halaman Detail Inventaris Pakan

Pada halaman inventaris pakan, dapat dilihat bahwa terdapat filter pakan yang berupa pakan industri, alami, dan Custom. Serta di bagian center terdapat dua bagian yang menampilkan list dari merk pakan dan pakan yang digunakan.

Dibagian Catat Pakan, terdapat form isian yang harus dilengkapi jika ingin mencatat pakan. Kemudian, pada halaman Edit Pakan memiliki layout yang kurang lebih sama seperti Catat Pakan namun fungsi yang digunakan berbeda.

2. Design route dan penerapan pada Flutter untuk inventaris suplemen (dalam bentuk RESTful API)

- (a) Design Sample Route

Berikut merupakan sample route yang sudah dibuat untuk inventaris

suplemen.

Group	Endpoint	HTTP Status	Operation	Purpose
Inventaris Suplemen	/inventory/suplemen?type=	GET	READ	mengambil semua data inventaris suplemen dengan filter type
	/inventory/suplemen/\${id}	GET	READ	mengambil data spesifik dari inventaris suplemen dengan id
	/inventory/suplemen	POST	CREATE	menginput data kedalam inventaris suplemen
	/inventory/suplemen/\${id}	PUT	UPDATE	memperbarui spesifik data pada inventaris suplemen dengan id
	/inventory/suplemen/\${id}	DELETE	DELETE	menghapus spesifik data pada inventaris suplemen dengan id

Gambar 4.39: Sample Route Inventaris Suplemen

(b) Model Backend

Berdasarkan sample route tersebut, dapat dibuat model dan class HTTP method pada backend. Berikut model serta class untuk inventaris suplemen :

- Model Inventaris Suplemen

```

1      class SuplemenInventory(db.Document):
2          id_int = db.SequenceField(required=True)
3          farm_id = db.ReferenceField(Farm, required=True)
4          function = db.StringField(required=True)
5          name = db.StringField(required=True)
6          description = db.StringField(required=True)
7          price = db.IntField(required=True)
8          amount = db.FloatField(required=True)
9          type = db.StringField(required=True)
10         min_expired_period = db.IntField(required=True)
11         max_expired_period = db.IntField(required=True)
12         image = db.StringField(required=True)
13         created_at = db.DateTimeField(default=datetime.datetime.
now())
14         updated_at = db.DateTimeField(default=datetime.datetime.
now())
15

```

(c) Fungsi-fungsi HTTP Method

- Mengambil semua data inventaris suplemen (HTTP Method - GET)

```

1      class SuplemenInventoriesApi(Resource):
2          @jwt_required()
3
4          def get(self):
5              try:
6                  current_user = get_jwt_identity()

```

```
7         farm = str(current_user['farm_id'])
8         farm_id = ObjectId(farm)
9
10        type = request.args.get('type') if request.args.get(
11            'type') else ""
12
13        name = request.args.get('name') if request.args.get(
14            'name') else ""
15
16        pipeline = [
17            {"$sort": {"id_int": 1}},
18            {
19                '$match': {
20                    "farm_id": farm_id,
21                    'function': {
22                        '$regex': type,
23                        '$options': 'i'
24                    }
25                },
26                {
27                    '$match': {
28                        'name': {
29                            '$regex': name,
30                            '$options': 'i'
31                        }
32                    }
33                }
34            }
35        ]
36
37        testing = SuplemenInventory.objects.aggregate(
38            pipeline)
39
40        temp = list(testing)
41
42        response = json.dumps({
43            'status': 'success',
44            'data': temp,
45        }, default=str)
46
47        return Response(response, mimetype="application/json",
48        ", status=200)
49
50        except Exception as e:
51            response = {"message": e}
52
53        response = json.dumps(response, default=str)
54
55        return Response(response, mimetype="application/json",
56        ", status=400)
57
```

- Mengambil spesifik data inventaris pakan (HTTP Method - GET)

```

1      class SuplemenInventoryApi(Resource):
2          def get(self, id):
3              try:
4                  pipeline = {"$match": {"id_int": int(id)}},
5                  testing = SuplemenInventory.objects.aggregate(
6                      pipeline)
7                  temp = list(testing)
8                  if len(temp) == 0:
9                      res = {"message": 'no data found'}
10                     response = json.dumps(res, default=str)
11                     return Response(response, mimetype="application/
12                         json", status=200)
13                     response = json.dumps({
14                         'status': 'success',
15                         'data': temp[0],
16                         }, default=str)
17                     return Response(response, mimetype="application/json
18                         ", status=200)
19                     except Exception as e:
20                         response = {"message": e}
21                         response = json.dumps(response, default=str)
22                         return Response(response, mimetype="application/json
23                         ", status=400)
24

```

- Menambahkan data inventaris pakan (HTTP Method - POST)

```

1      class SuplemenInventoriesApi(Resource):
2          @jwt_required()
3          def post(self):
4              try:
5                  current_user = get_jwt_identity()
6                  farm = str(current_user['farm_id'])
7                  body = {
8                      "farm_id": farm,
9                      "function": request.form.get('function', None),
10                     "name": request.form.get('name', None),
11                     "description": request.form.get('description',
12                         None),
12                     "price": request.form.get('price', None),
13                     "amount": request.form.get('amount', None),
14

```

```

14         "type": request.form.get('type', None),
15         "min_expired_period": request.form.get(
16             'min_expired_period', None),
17             "max_expired_period": request.form.get(
18                 'max_expired_period', None),
19                 "image": request.form.get('image', None)
20             }
21             inventory = SuplemenInventory(**body).save()
22             id = inventory.id
23             res = {"message": "success add suplemen to"
24             inventory", "id": id, "data": body}
25             response = json.dumps(res, default=str)
26             return Response(response, mimetype="application/
27             json", status=200)
28         except Exception as e:
29             response = {"message": str(e)}
30             response = json.dumps(response, default=str)
31             return Response(response, mimetype="application/
32             json", status=400)
33

```

- Memperbarui spesifik data inventaris pakan (HTTP Method - PUT)

```

1     class SuplemenInventoryApi(Resource):
2         def put(self, id):
3             try:
4                 body = {
5                     "id_int": int(id),
6                     "function": request.form.get('function', None),
7                     "name": request.form.get('name', None),
8                     "description": request.form.get('description',
None),
9                     "price": request.form.get('price', None),
10                    "amount": request.form.get('amount', None),
11                    "type": request.form.get('type', None),
12                    "min_expired_period": request.form.get(
13                        'min_expired_period', None),
14                        "max_expired_period": request.form.get(
15                            'max_expired_period', None),
16                            "image": request.form.get('image', None)
17                        }
18                        inventory = SuplemenInventory.objects.get(id_int =
int(id)).update(**body)
19                        response = {"message": "success update suplemen"
20

```

```

    inventory", "data": body}
18         response = json.dumps(response, default=str)
19         return Response(response, mimetype="application/json
", status=200)
20     except Exception as e:
21         response = {"message": str(e)}
22         response = json.dumps(response, default=str)
23         return Response(response, mimetype="application/json
", status=400)
24

```

- Menghapus spesifik data inventaris pakan (HTTP Method - DELETE)

```

1      class SuplemenInventoryApi(Resource):
2          def delete(self, id):
3              try:
4                  inventory = SuplemenInventory.objects.get(id=int =
int(id)).delete()
5                  response = {"message": "success delete suplemen
inventory"}
6                  response = json.dumps(response, default=str)
7                  return Response(response, mimetype="application/json
", status=200)
8              except Exception as e:
9                  response = {"message": str(e)}
10                 response = json.dumps(response, default=str)
11                 return Response(response, mimetype="application/json
", status=400)
12

```

(d) Controller HTTP pada Flutter

- Mengambil semua data inventaris suplemen (HTTP Method - GET)

```

1      Future getAllData(String type, Function() doAfter) async {
2          suplemenList.value.data!.clear();
3          isLoadingPage.value = true;
4          isProbLoading.value = true;
5          isCarbLoading.value = true;
6          listCarbon.clear();
7          listCultureProbiotik.clear();
8

```

```

9         SharedPreferences prefs = await SharedPreferences.
10        getInstance();
11        String token = prefs.getString('token').toString();
12        var headers = {'Authorization': 'Bearer $token'};
13
14        final response = await http.get(
15          Uri.parse('${Urls.invSup}?type=$type'),
16          headers: headers,
17        );
18
19        try {
20          if (response.statusCode == 200) {
21            InventarisSuplemenModel res =
22              InventarisSuplemenModel.fromJson(jsonDecode(response
23                .body));
24
25            suplemenList.value = res;
26            // inspect(suplemenList.value.data);
27
28            if (suplemenList.value.data!.isNotEmpty) {
29              for (var i in suplemenList.value.data!) {
30                if (type == 'Probiotik') {
31                  listCultureProbiotik.add({
32                    'id': i.idInt,
33                    'suplemen_id': i.sId,
34                    'suplemen_name': i.name,
35                  });
36
37                  selectedCultureProbiotik.value =
38                    listCultureProbiotik[0];
39
40                  if (type == 'Feed Additive') {
41                    listCarbon.add({
42                      'id': i.idInt,
43                      'suplemen_id': i.sId,
44                      'suplemen_name': i.name,
45                    });
46
47                  selectedCarbon.value = listCarbon[0];
48
49                  // inspect(listCultureProbiotik);

```

```

50
51         doAfter();
52     }
53     } catch (e) {
54     inspect(e);
55     throw Exception(e);
56   }
57   isProbLoading.value = false;
58   isCarbLoading.value = false;
59   isLoadingPage.value = false;
60 }
61

```

- Mengambil spesifik data inventaris suplemen (HTTP Method - GET)

```

1      Future getDataByID(int id, Function() doAfter) async {
2          isLoadingDetail.value = true;
3
4          final response = await http.get(Uri.parse('${Urls.invSup
5              }/$id'));
6
7          try {
8              if (response.statusCode == 200) {
9                  DetailInventarissuplemenModel res =
10                     DetailInventarissuplemenModel.fromJson(jsonDecode(
11                         response.body));
12
13                     // for (var i in listSuplemenName) {
14                     //   if (i['suplemen_name_id'] == res.data.fee)
15                     // }
16
17                     functionCategory.value = res.data!.function!.toString
18                     ();
19                     name.text = res.data!.name!.toString();
20                     selectedFeedAdditive.value = res.data!.name!.toString
21                     ();
22                     desc.text = res.data!.description.toString();
23                     price.text = res.data!.price.toString();
24                     amount.text = res.data!.amount!.toStringAsFixed(2);
25                     typeCategory.value = res.data!.type.toString();
26                     minExp.text = res.data!.minExpiredPeriod.toString();
27                     maxExp.text = res.data!.maxExpiredPeriod.toString();
28                     image.value = res.data!.image.toString();
29               }
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
317
318
319
319
320
321
322
323
324
325
326
327
327
328
329
329
330
331
332
333
334
335
335
336
337
337
338
339
339
340
341
342
343
344
344
345
346
346
347
347
348
349
349
350
351
351
352
353
353
354
354
355
356
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
14
```

```

26         doAfter();
27     } catch (e) {
28         throw Exception(e);
29     }
30     isLoadingDetail.value = false;
31 }
32

```

- Menambahkan data inventaris suplemen (HTTP Method - POST)

```

1      Future postData(Function() doAfter) async {
2          var map = <String, dynamic>{};
3
4          SharedPreferences prefs = await SharedPreferences.
5              getInstance();
6          String token = prefs.getString('token').toString();
7          var headers = {'Authorization': 'Bearer $token'};
8
9          map['function'] = functionCategory.value;
10         map['name'] = functionCategory.value == 'Feed Additive'
11             ? selectedFeedAdditive.value
12             : name.text;
13         map['description'] = desc.text == '' ? '-' : desc.text;
14         map['price'] = price.text;
15         map['amount'] = amount.text.replaceAll(',', '.');
16         map['type'] = typeCategory.value;
17         map['min_expired_period'] = minExp.text == '' ? '0' :
18             minExp.text;
19         map['max_expired_period'] = maxExp.text == ''
20             ? (int.parse(minExp.text == '' ? '0' : minExp.text) *
21                 2).toString()
22             : maxExp.text;
23         map['image'] = image.value;
24
25         isLoadingPost.value = true;
26
27         inspect(map);
28
29         try {
30             await http.post(
31                 Uri.parse(Urls.invSup),
32                 body: map,
33                 headers: headers,
34             );
35         }
36     }
37
38     doAfter();
39 }
40

```

```

32         doAfter();
33     } catch (e) {
34         throw Exception(e);
35     }
36     isLoadingPost.value = false;
37 }
38
39

```

- Memperbarui spesifik data inventaris suplemen (HTTP Method - PUT)

```

1     Future updateData(int id, Function() doAfter) async {
2         var map = <String, dynamic>{};
3
4         inspect(id);
5
6         map['function'] = functionCategory.value;
7         map['name'] = functionCategory.value == 'Feed Additive'
8             ? selectedFeedAdditive.value
9             : name.text;
10        map['description'] = desc.text == '' ? '-' : desc.text;
11        map['price'] = price.text;
12        map['amount'] = amount.text.replaceAll(',', '.');
13        map['type'] = typeCategory.value;
14        map['min_expired_period'] = minExp.text == '' ? '0' :
minExp.text;
15        map['max_expired_period'] = maxExp.text == ''
16            ? (int.parse(minExp.text == '' ? '0' : minExp.text) *
2).toString()
17            : maxExp.text;
18        map['image'] = image.value;
19
20        isLoadingPost.value = true;
21
22        try {
23            inspect(map);
24            final res = await http.put(
25                Uri.parse('${Urls.invSup}/$id'),
26                body: map,
27            );
28            inspect(res);
29            doAfter();
30        } catch (e) {

```

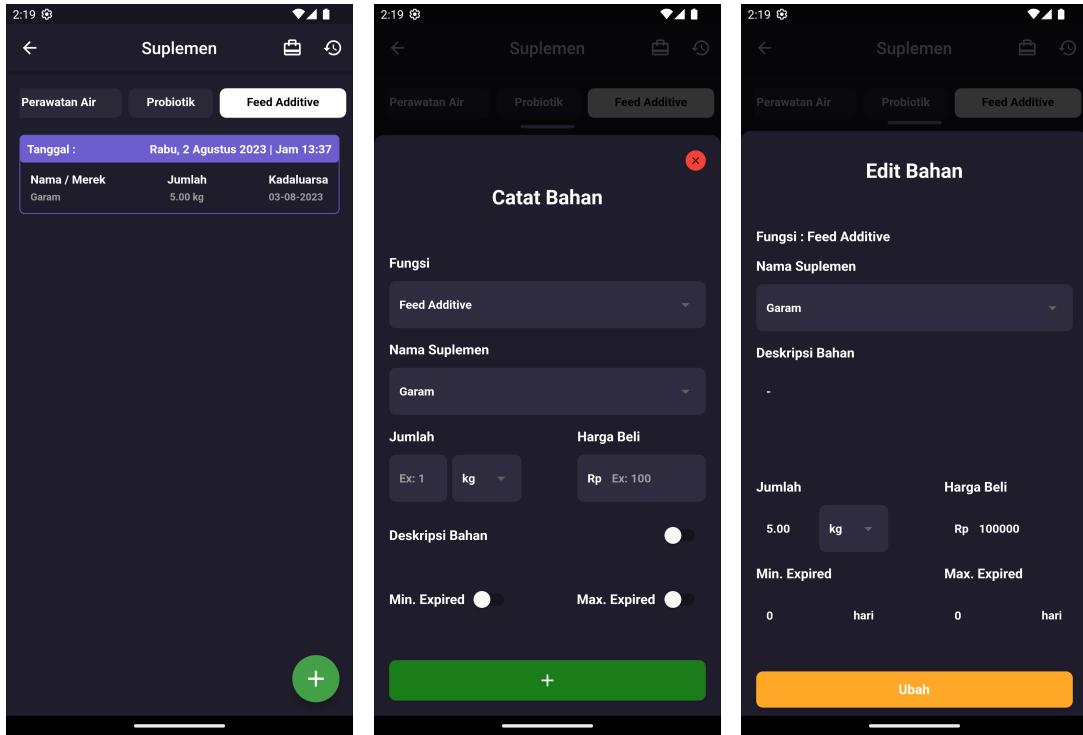
```
31         throw Exception(e);
32     }
33     isLoadingPost.value = false;
34 }
35 }
```

- Menghapus spesifik data inventaris suplemen (HTTP Method - DELETE)

```
1     Future deleteData(int id, Function() doAfter) async {
2         isLoadingDelete.value = true;
3         try {
4             await http.delete(
5                 Uri.parse(
6                     '${Urls.invSup}/$id',
7                 ),
8                 headers: {
9                     'Content-Type': 'application/json; charset=UTF-8',
10                },
11            );
12            doAfter();
13        } catch (e) {
14            throw Exception(e);
15        }
16        isLoadingDelete.value = false;
17    }
18 }
```

(e) Tampilan pada Flutter

- Halaman Inventaris Suplemen, Input Suplemen, dan Detail Suplemen



Gambar 4.40: Halaman Inventaris Suplemen

Gambar 4.41: Halaman Input Inventaris Suplemen

Gambar 4.42: Halaman Detail Inventaris Suplemen

Pada Halaman inventaris suplemen, dapat dilihat bahwa terdapat beberapa filter suplemen seperti Feed Additive, Probiotik, Perawatan Air, dan Obat . Serta di bagian center terdapat list dari inventaris suplemen yang sudah terdaftar.

Dibagian Catat suplemen, terdapat form isian yang harus dilengkapi jika ingin mencatat suplemen. Kemudian, pada halaman Edit Suplemen memiliki layout yang kurang lebih sama seperti Catat Suplemen namun fungsi yang digunakan berbeda.

3. Design route dan penerapan pada Flutter untuk inventaris listrik (dalam bentuk RESTful API)

(a) Design Sample Route

Berikut merupakan sample route yang sudah dibuat untuk inventaris

pakan.

Group	Endpoint	HTTP Status	Operation	Purpose
Inventaris Listrik	/inventory/electric?type=&start_date=&end_date=	GET	READ	mengambil semua data inventaris listrik dengan filter tipe dan tanggal
	/inventory/electric/\${id}	GET	READ	mengambil data spesifik dari inventaris listrik dengan id
	/inventory/electric	POST	CREATE	menginput data kedalam inventaris listrik
	/inventory/electric/\${id}	PUT	UPDATE	memperbarui spesifik data pada inventaris listrik dengan id
	/inventory/electric/\${id}	DELETE	DELETE	menghapus spesifik data pada inventaris listrik dengan id

Gambar 4.43: Sample Route Inventaris Listrik

(b) Model Backend

Berdasarkan sample route tersebut, dapat dibuat model dan class HTTP method pada backend. Berikut model serta class untuk inventaris listrik :

- Model Inventaris Listrik

```

1      class ElectricInventory(db.Document):
2          id_int = db.SequenceField(required=True)
3          farm_id = db.ReferenceField(Farm, required=True)
4          type = db.StringField(required=True)
5          name = db.StringField(required=True)
6          daya = db.StringField()
7          price = db.IntField(required=True)
8          id_token = db.StringField()
9          month = db.StringField()
10         image = db.StringField(required=True)
11         created_at = db.DateTimeField(default=datetime.datetime.
now)
12         updated_at = db.DateTimeField(default=datetime.datetime.
now)
13

```

(c) Fungsi-fungsi HTTP Method

- Mengambil semua data inventaris listrik (HTTP Method - GET)

```

1      class ElectricInventoriesApi(Resource):
2          @jwt_required()
3
4          def get(self):
5              try:
6                  current_user = get_jwt_identity()
7                  farm = str(current_user['farm_id'])
8                  farm_id = ObjectId(farm)
9

```

```

10         start_date = datetime.datetime.strptime(request.args
11             .get('start_date'), '%Y-%m-%d') if request.args.get('start_date'
12             ) else datetime.datetime.strptime("2023-01-01", '%Y-%m-%d')
13             end_date = datetime.datetime.strptime(request.args.
14                 get('end_date'), '%Y-%m-%d') + datetime.timedelta(days=1) if
15                 request.args.get('end_date') else datetime.datetime.strptime(""
16                     "2030-01-01", '%Y-%m-%d')
17             type = request.args.get('type') if request.args.get(
18                 'type') else ""
19
20             pipeline = [
21                 {"$sort": {"id_int": 1}},
22                 {
23                     '$match': {
24                         'created_at': {
25                             '$gte': start_date,
26                             '$lte': end_date,
27                         }
28                     }
29                 },
30                 {
31                     '$match': {
32                         "farm_id": farm_id,
33                         'type': {
34                             '$regex': type,
35                             '$options': 'i'
36                         }
37                     }
38                 }
39             ]
40
41             testing = ElectricInventory.objects.aggregate(
42                 pipeline)
43             temp = list(testing)
44             response = json.dumps({
45                 'status': 'success',
46                 'data': temp,
47             }, default=str)
48             return Response(response, mimetype="application/json
49             ", status=200)
50             except Exception as e:
51                 response = {"message": e}
52                 response = json.dumps(response, default=str)
53                 return Response(response, mimetype="application/json
54             "
55         
```

```
    ", status=400)
46
```

- Mengambil spesifik data inventaris listrik (HTTP Method - GET)

```
1      class ElectricInventoryApi(Resource):
2          def get(self, id):
3              try:
4                  pipeline = {"$match": {"id_int": int(id)}},
5                  testing = ElectricInventory.objects.aggregate(
6                      pipeline)
7                  temp = list(testing)
8                  if len(temp) == 0:
9                      res = {"message": 'no data found'}
10                     response = json.dumps(res, default=str)
11                     return Response(response, mimetype="application/
12                         json", status=200)
13                     response = json.dumps({
14                         'status': 'success',
15                         'data': temp[0],
16                         }, default=str)
17                     return Response(response, mimetype="application/json
18                         ", status=200)
19                     except Exception as e:
20                         response = {"message": e}
21                         response = json.dumps(response, default=str)
22                         return Response(response, mimetype="application/json
23                         ", status=400)
24
```

- Menambahkan data inventaris listrik (HTTP Method - POST)

```
1      class ElectricInventoriesApi(Resource):
2          @jwt_required()
3          def post(self):
4              try:
5                  current_user = get_jwt_identity()
6                  farm = str(current_user['farm_id'])
7                  body = {
8                      "farm_id": farm,
9                      "name": request.form.get('name', None),
10                     "price": request.form.get('price', None),
11                     "type": request.form.get('type', None),
12                     "daya": request.form.get('daya', None),
13                     "image": request.form.get('image', None),
```

```
14             "id_token": request.form.get('id_token', None),
15             "month": request.form.get('month', None)
16         }
17
18         inventory = ElectricInventory(**body).save()
19
20         id = inventory.id
21
22         res = {"message": "success add electric to inventory",
23                "id": id, "data": body}
24
25         response = json.dumps(res, default=str)
26
27         return Response(response, mimetype="application/json",
28                         status=200)
29
30     except Exception as e:
31
32         response = {"message": str(e)}
33
34         response = json.dumps(response, default=str)
35
36         return Response(response, mimetype="application/json",
37                         status=400)
```

- Memperbarui spesifik data inventaris listrik (HTTP Method - PUT)

```
1 class ElectricInventoryApi(Resource):
2
3     def put(self, id):
4
5         try:
6
7             body = {
8
9                 "id_int": int(id),
10
11                 "name": request.form.get('name', None),
12
13                 "price": request.form.get('price', None),
14
15                 "type": request.form.get('type', None),
16
17                 "daya": request.form.get('daya', None),
18
19                 "image": request.form.get('image', None),
20
21                 "id_token": request.form.get('id_token', None),
22
23                 "month": request.form.get('month', None)
24
25             }
26
27             inventory = ElectricInventory.objects.get(id_int =
28
29                 int(id)).update(**body)
30
31             response = {"message": "success update electric
32
33                 inventory", "data": body}
34
35             response = json.dumps(response, default=str)
36
37             return Response(response, mimetype="application/json
38
39                 ", status=200)
40
41             except Exception as e:
42
43                 response = {"message": str(e)}
44
45                 response = json.dumps(response, default=str)
46
47                 return Response(response, mimetype="application/json
48
49                 ", status=200)
```

```

    ", status=400)
23

```

- Menghapus spesifik data inventaris listrik (HTTP Method - DELETE)

```

1      class ElectricInventoryApi(Resource):
2          def delete(self, id):
3              try:
4                  inventory = ElectricInventory.objects.get(id=int =
int(id)).delete()
5                  response = {"message": "success delete electric
inventory"}
6                  response = json.dumps(response, default=str)
7                  return Response(response, mimetype="application/json
", status=200)
8              except Exception as e:
9                  response = {"message": str(e)}
10                 response = json.dumps(response, default=str)
11                 return Response(response, mimetype="application/json
", status=400)
12

```

(d) Controller HTTP pada Flutter

- Mengambil semua data inventaris listrik (HTTP Method - GET)

```

1      Future getAllData(
2          String first, String last, String type, Function()
doAfter) async {
3          electricList.value.data!.clear();
4          isLoadingPage.value = true;
5
6          SharedPreferences prefs = await SharedPreferences.
getInstance();
7          String token = prefs.getString('token').toString();
8          var headers = {'Authorization': 'Bearer $token'};
9
10         final response = await http.get(
11             Uri.parse('${Urls.invElect}?type=$type&first=$first&
last=$last'),
12             headers: headers);
13
14         try {

```

```

15         if (response.statusCode == 200) {
16             InventarisListrikModel res =
17                 InventarisListrikModel.fromJson(jsonDecode(response.
18                     body));
19
19             electricList.value = res;
20             inspect(electricList.value.data);
21             doAfter();
22         }
23     } catch (e) {
24         throw Exception(e);
25     }
26     isLoadingPage.value = false;
27 }
28

```

- Mengambil spesifik data inventaris listrik (HTTP Method - GET)

```

1         Future getDataByID(int id, Function() doAfter) async {
2             isLoadingDetail.value = true;
3
4             final response = await http.get(Uri.parse('${Urls.
5                 invElect}/$id'));
6
7             try {
8                 if (response.statusCode == 200) {
9                     DetailInventarisListrikModel res =
10                         DetailInventarisListrikModel.fromJson(jsonDecode(
11                             response.body));
12
12                     electricCategory.value = res.data!.type.toString();
13                     name.text = res.data!.name.toString();
14                     power.text = res.data!.daya.toString();
15                     price.text = res.data!.price.toString();
16                     idToken.text = res.data!.idToken.toString();
17                     monthPicked.text = res.data!.month.toString();
18                     image.value = res.data!.image.toString();
19                 }
20                 doAfter();
21             } catch (e) {
22                 throw Exception(e);
23             }
24             isLoadingDetail.value = false;
25         }

```

25

- Menambahkan data inventaris listrik (HTTP Method - POST)

```

1      Future postData(Function() doAfter) async {
2          var map = <String, dynamic>{};
3
4          SharedPreferences prefs = await SharedPreferences.
5              getInstance();
6          String token = prefs.getString('token').toString();
7          var headers = {'Authorization': 'Bearer $token'};
8
9          map['name'] = name.text;
10         map['type'] = electricCategory.value;
11         map['price'] = price.text;
12         map['daya'] = power.text;
13         map['image'] = image.value;
14         map['id_token'] = idToken.text;
15         map['month'] = monthPicked.text;
16
17         isLoadingPost.value = true;
18
19         try {
20             await http.post(
21                 Uri.parse(Urls.invElect),
22                 body: map,
23                 headers: headers,
24             );
25             doAfter();
26         } catch (e) {
27             throw Exception(e);
28         }
29         isLoadingPost.value = false;
30     }
31

```

- Memperbarui spesifik data inventaris listrik (HTTP Method - PUT)

```

1      Future updateData(int id, Function() doAfter) async {
2          var map = <String, dynamic>{};
3
4          map['name'] = name.text;
5          map['type'] = electricCategory.value;
6          map['price'] = price.text;

```

```

7      map['daya'] = power.text;
8      map['image'] = image.value;
9      map['id_token'] = idToken.text;
10     map['month'] = monthPicked.text;
11
12     isLoadingPost.value = true;
13
14     try {
15       inspect(map);
16       await http.put(
17         Uri.parse('${Urls.invElect}/$id'),
18         body: map,
19       );
20       doAfter();
21     } catch (e) {
22       throw Exception(e);
23     }
24     isLoadingPost.value = false;
25   }
26

```

- Menghapus spesifik data inventaris listrik (HTTP Method - DELETE)

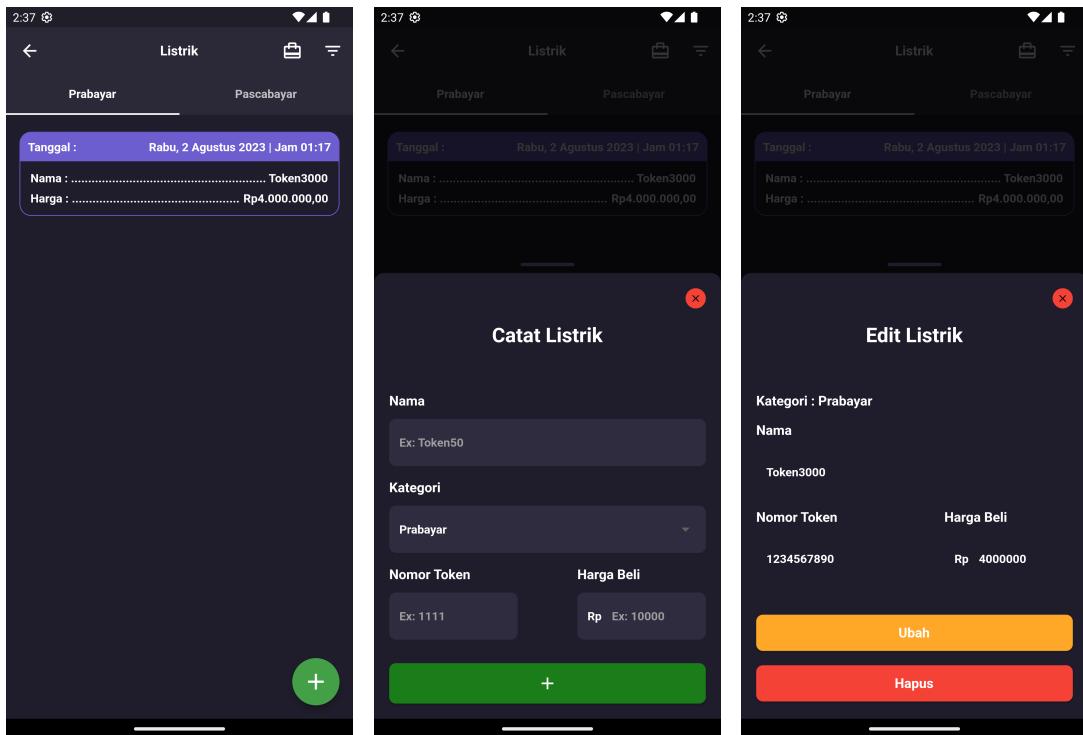
```

1      Future deleteData(int id, Function() doAfter) async {
2        isLoadingDelete.value = true;
3        try {
4          await http.delete(
5            Uri.parse(
6              '${Urls.invElect}/$id',
7            ),
8            headers: {
9              'Content-Type': 'application/json; charset=UTF-8',
10             },
11           );
12           doAfter();
13         } catch (e) {
14           throw Exception(e);
15         }
16         isLoadingDelete.value = false;
17       }
18

```

(e) Tampilan pada Flutter

- Halaman Inventaris Listrik, Input Listrik, dan Detail Listrik



Gambar 4.44: Halaman Inventaris Listrik

Gambar 4.45: Halaman Input Inventaris Listrik

Gambar 4.46: Halaman Detail Inventaris Listrik

Pada tampilan tersebut, dapat dilihat bahwa terdapat filter listrik yang berupa pakan prabayar dan pascabayar, masing-masing memiliki input form yang berbeda. Serta di bagian center terdapat bagian yang menampilkan list dari inventaris listrik yang sudah terdaftar.

Dibagian Catat Listrik, terdapat form isian yang harus dilengkapi jika ingin mencatat inventaris listrik. Kemudian, pada halaman Edit Listrik memiliki layout yang kurang lebih sama seperti Catat Listrik namun fungsi yang digunakan berbeda.

4. Design route dan penerapan pada Flutter untuk inventaris aset (dalam bentuk RESTful API)

(a) Design Sample Route

Berikut merupakan sample route yang sudah dibuat untuk inventaris aset.

Group	Endpoint	HTTP Status	Operation	Purpose
Inventaris Aset	/inventory/asset?type=&start_date=&end_date=	GET	READ	mengambil semua data inventaris aset dengan filter tipe dan tanggal
	/inventory/asset/\${id}	GET	READ	mengambil data spesifik dari inventaris aset dengan id
	/inventory/asset	POST	CREATE	menginput data kedalam inventaris aset
	/inventory/asset/\${id}	PUT	UPDATE	memperbarui spesifik data pada inventaris aset dengan id
	/inventory/asset/\${id}	DELETE	DELETE	menghapus spesifik data pada inventaris aset dengan id

Gambar 4.47: Sample Route Inventaris Aset

(b) Model Backend

Berdasarkan sample route tersebut, dapat dibuat model dan class HTTP method pada backend. Berikut model serta class untuk inventaris aset :

- Model Inventaris Aset

```

1      class AssetInventory(db.Document):
2          id_int = db.SequenceField(required=True)
3          farm_id = db.ReferenceField(Farm, required=True)
4          asset_category = db.StringField(required=True)
5          name = db.StringField(required=True)
6          description = db.StringField(required=True)
7          amount = db.IntField(required=True)
8          price = db.IntField(required=True)
9          image = db.StringField(required=True)
10         created_at = db.DateTimeField(default=datetime.datetime.
11                                         now)
11         updated_at = db.DateTimeField(default=datetime.datetime.
12                                         now)

```

(c) Fungsi-fungsi HTTP Method

- Mengambil semua data inventaris aset (HTTP Method - GET)

```

1      class AssetInventoriesApi(Resource):
2          @jwt_required()
3          def get(self):
4              try:
5                  current_user = get_jwt_identity()
6                  farm = str(current_user['farm_id'])
7                  farm_id = ObjectId(farm)
8

```

```
9         type = request.args.get('type') if request.args.get('type') else ""
10        start_date = datetime.datetime.strptime(request.args.get('start_date'), '%Y-%m-%d') if request.args.get('start_date') else datetime.datetime.strptime("2023-01-01", '%Y-%m-%d')
11        end_date = datetime.datetime.strptime(request.args.get('end_date'), '%Y-%m-%d') + datetime.timedelta(days=1) if request.args.get('end_date') else datetime.datetime.strptime("2030-01-01", '%Y-%m-%d')
12
13        pipeline = [
14            {"$sort": {"id_int": 1}},
15            {
16                '$match': {
17                    "farm_id": farm_id,
18                    'asset_category': {
19                        '$regex': type,
20                        '$options': 'i'
21                    },
22                    'created_at': {
23                        '$gte': start_date,
24                        '$lte': end_date,
25                    }
26                }
27            },
28        ]
29
30        testing = AssetInventory.objects.aggregate(pipeline)
31        temp = list(testing)
32        response = json.dumps({
33            'status': 'success',
34            'data': temp,
35        }, default=str)
36        return Response(response, mimetype="application/json",
37        ", status=200)
38    except Exception as e:
39        response = {"message": e}
40        response = json.dumps(response, default=str)
41        return Response(response, mimetype="application/json",
42        ", status=400)
```

- Mengambil spesifik data inventaris aset (HTTP Method - GET)

```

1      class AssetInventoryApi(Resource):
2          def get(self, id):
3              try:
4                  pipeline = {"$match": {"id_int": int(id)}},
5                  testing = AssetInventory.objects.aggregate(pipeline)
6                  temp = list(testing)
7                  if len(temp) == 0:
8                      res = {"message": 'no data found'}
9                      response = json.dumps(res, default=str)
10                     return Response(response, mimetype="application/
11 json", status=200)
12                     response = json.dumps({
13                         'status': 'success',
14                         'data': temp[0],
15                         }, default=str)
16                     return Response(response, mimetype="application/json
17 ", status=200)
18                     except Exception as e:
19                         response = {"message": e}
20                         response = json.dumps(response, default=str)
21                         return Response(response, mimetype="application/json
22 ", status=400)
23

```

- Menambahkan data inventaris aset (HTTP Method - POST)

```

1      class AssetInventoriesApi(Resource):
2          @jwt_required()
3          def post(self):
4              try:
5                  current_user = get_jwt_identity()
6                  farm = str(current_user['farm_id'])
7                  body = {
8                      "farm_id": farm,
9                      "asset_category": request.form.get('asset_category
10 , None),
11                      "name": request.form.get('name', None),
12                      "description": request.form.get('description',
13 None),
14                      "amount": request.form.get('amount', None),
15                      "price": request.form.get('price', None),
16                      "image": request.form.get('image', None),
17                  }
18                  inventory = AssetInventory(**body).save()
19

```

```

17         id = inventory.id
18
19             res = {"message": "success add asset to inventory",
20 "id": id, "data": body}
21
22             response = json.dumps(res, default=str)
23
24             return Response(response, mimetype="application/json
25 ", status=200)
26
27     except Exception as e:
28
29         response = {"message": str(e)}
30
31         response = json.dumps(response, default=str)
32
33         return Response(response, mimetype="application/json
34 ", status=400)
35
36

```

- Memperbarui spesifik data inventaris aset (HTTP Method - PUT)

```

1     class AssetInventoryApi(Resource):
2
3         def put(self, id):
4
5             try:
6
7                 body = {
8
9                     "id_int": int(id),
10
11                     "asset_category": request.form.get('asset_category
12
13                     ', None),
14
15                     "name": request.form.get('name', None),
16
17                     "description": request.form.get('description',
18
19                     None),
20
21                     "amount": request.form.get('amount', None),
22
23                     "price": request.form.get('price', None),
24
25                     "image": request.form.get('image', None),
26
27                 }
28
29                 inventory = AssetInventory.objects.get(id_int = int(
30
31                     id)).update(**body)
32
33                 response = {"message": "success update asset
34
35                     inventory", "data": body}
36
37                 response = json.dumps(response, default=str)
38
39                 return Response(response, mimetype="application/json
40
41                     ", status=200)
42
43             except Exception as e:
44
45                 response = {"message": str(e)}
46
47                 response = json.dumps(response, default=str)
48
49                 return Response(response, mimetype="application/json
50
51                     ", status=400)
52
53

```

- Menghapus spesifik data inventaris aset (HTTP Method - DELETE)

```
1         class AssetInventoryApi(Resource):
2             def delete(self, id):
3                 try:
4                     inventory = AssetInventory.objects.get(id=int(id)).delete()
5                     response = {"message": "success delete asset inventory"}
6                     response = json.dumps(response, default=str)
7                     return Response(response, mimetype="application/json",
8                                     status=200)
9                 except Exception as e:
10                     response = {"message": str(e)}
11                     response = json.dumps(response, default=str)
12                     return Response(response, mimetype="application/json",
13                                     status=400)
```

(d) Controller HTTP pada Flutter

- Mengambil semua data inventaris aset (HTTP Method - GET)

```
1         Future getAllData(
2             String type, String first, String last, Function()
3             doAfter) async {
4                 assetList.value.data!.clear();
5                 isLoadingPage.value = true;
6
7                 SharedPreferences prefs = await SharedPreferences.
8                 getInstance();
9
10                String token = prefs.getString('token').toString();
11                var headers = {'Authorization': 'Bearer $token'};
12
13                final response = await http.get(
14                    Uri.parse('${Urls.invAsset}?type=$type&start_date=$first
15 &end_date=$last'),
16                    headers: headers,
17                );
18
19
20                try {
21                    if (response.statusCode == 200) {
22                        InventarisAssetModel res =
23                            InventarisAssetModel.fromJson(jsonDecode(response.
24 body));
25
26                    }
27
28                } catch (e) {
29                    print(e);
30                }
31            }
32        }
33    }
34
35
```

```

20         assetList.value = res;
21
22         doAfter();
23     }
24 } catch (e) {
25     throw Exception(e);
26 }
27 isLoadingPage.value = false;
28 }
29

```

- Mengambil spesifik data inventaris aset (HTTP Method - GET)

```

1     Future getDataByID(int id, Function() doAfter) async {
2
3         isLoadingDetail.value = true;
4
5         final response = await http.get(Uri.parse('${Urls.
6             invAsset}/$id'));
7
8         try {
9             if (response.statusCode == 200) {
10                 DetailInventarisAssetModel res =
11                     DetailInventarisAssetModel.fromJson(jsonDecode(
12                     response.body));
13
14                 assetCategory.value = res.data!.assetCategory!.to
15                 string();
16                 name.text = res.data!.name.toString();
17                 desc.text = res.data!.description.toString() == ''
18                 ? '-'
19                 : res.data!.description.toString();
20                 price.text = res.data!.price.toString();
21                 amount.text = res.data!.amount.toString();
22                 image.value = res.data!.image.toString();
23             }
24             doAfter();
25         } catch (e) {
26             throw Exception(e);
27         }
28         isLoadingDetail.value = false;
29     }
30

```

- Menambahkan data inventaris aset (HTTP Method - POST)

```

1     Future postData(Function() doAfter) async {
2         var map = <String, dynamic>{};
3
4         SharedPreferences prefs = await SharedPreferences.
5             getInstance();
6         String token = prefs.getString('token').toString();
7         var headers = {'Authorization': 'Bearer $token'};
8         map['asset_category'] = assetCategory.value;
9         map['name'] = name.text;
10        map['description'] = desc.text;
11        map['price'] = price.text;
12        map['amount'] = amount.text;
13        map['image'] = image.value;
14
15        isLoadingPost.value = true;
16
17        try {
18            final res = await http.post(
19                Uri.parse(Urls.invAsset),
20                body: map,
21                headers: headers,
22            );
23            inspect(res);
24            doAfter();
25        } catch (e) {
26            throw Exception(e);
27        }
28        isLoadingPost.value = false;
29    }

```

- Memperbarui spesifik data inventaris aset (HTTP Method - PUT)

```

1     Future updateData(int id, Function() doAfter) async {
2         var map = <String, dynamic>{};
3
4         map['asset_category'] = assetCategory.value;
5         map['name'] = name.text;
6         map['description'] = desc.text == '' ? '-' : desc.text;
7         map['price'] = price.text;
8         map['amount'] = amount.text;
9         map['image'] = image.value;
10
11        isLoadingPost.value = true;

```

```

12
13     try {
14         // inspect(map);
15         final res = await http.put(
16             Uri.parse('${Urls.invAsset}/$id'),
17             body: map,
18         );
19         // inspect(res);
20         doAfter();
21     } catch (e) {
22         throw Exception(e);
23     }
24     isLoadingPost.value = false;
25 }
26

```

- Menghapus spesifik data inventaris aset (HTTP Method - DELETE)

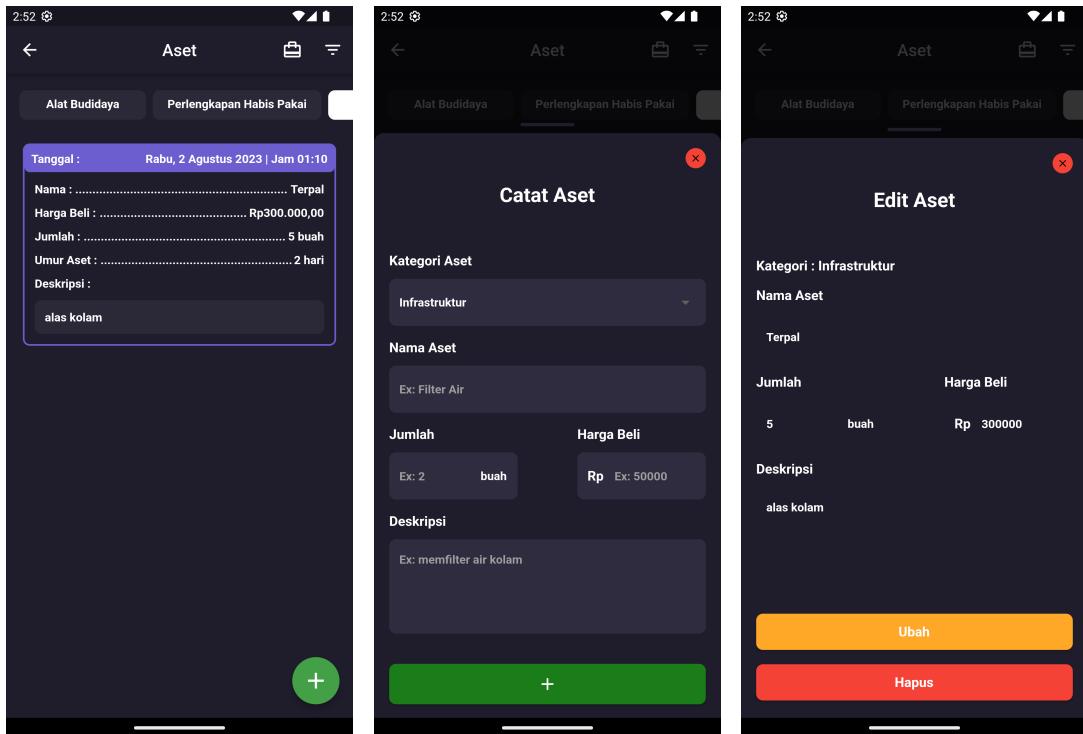
```

1     Future deleteData(int id, Function() doAfter) async {
2         isLoadingDelete.value = true;
3         try {
4             await http.delete(
5                 Uri.parse(
6                     '${Urls.invAsset}/$id',
7                 ),
8                 headers: {
9                     'Content-Type': 'application/json; charset=UTF-8',
10                },
11            );
12            doAfter();
13        } catch (e) {
14            throw Exception(e);
15        }
16        isLoadingDelete.value = false;
17    }
18

```

(e) Tampilan pada Flutter

- Halaman Inventaris Aset, Input Aset, dan Detail Aset



Gambar 4.48: Halaman Inventaris Aset

Gambar 4.49: Halaman Input Inventaris Aset

Gambar 4.50: Halaman Detail Inventaris Aset

Pada tampilan tersebut, dapat dilihat bahwa terdapat filter pakan yang berupa aset alat budidaya, aset perlengkapan habis pakai, aset infrastruktur, dan lainnya. Serta di bagian center terdapat bagian yang menampilkan list dari inventaris aset yang sudah terdaftar.

Dibagian Catat Aset, terdapat form isian yang harus dilengkapi jika ingin mencatat Aset. Kemudian, pada halaman Edit Aset memiliki layout yang kurang lebih sama seperti Catat Aset namun fungsi yang digunakan berbeda.

5. Design route dan penerapan pada Flutter untuk riwayat pemakaian pakan (dalam bentuk RESTful API)

(a) Design Sample Route

Berikut merupakan sample route yang sudah dibuat untuk riwayat

pemakaian pakan.

Group	Endpoint	HTTP Status	Operation	Purpose
History Inventaris Pakan	/history/inventory/feed?start_date=&end_date=&pond_name=&name=	GET	READ	mengambil data riwayat penggunaan pakan
	/history/inventory/feed	POST	CREATE	menginput data penggunaan pakan

Gambar 4.51: Sample Route Riwayat Pemakaian Pakan

(b) Model Backend

Berdasarkan sample route tersebut, dapat dibuat model dan class HTTP method pada backend. Berikut model serta class untuk riwayat pemakaian pakan :

- Model Riwayat Pemakaian Pakan

```

1      class FeedUsed(db.Document):
2          fish_feed_id = db.ReferenceField(FeedInventory, required
3          =True)
4          farm_id = db.ReferenceField(Farm, required=True)
5          original_amount = db.FloatField(required=True)
6          usage = db.FloatField(required=True)
7          pond = db.StringField(required=True)
8          created_at = db.DateTimeField(default=datetime.datetime.
now())
9          updated_at = db.DateTimeField(default=datetime.datetime.
now())

```

(c) Fungsi-fungsi HTTP Method

- Mengambil semua data riwayat pemakaian pakan (HTTP Method - GET)

```

1      class FeedFishHistoryApi(Resource):
2          @jwt_required()
3
4          def get(self):
5              try:
6                  current_user = get_jwt_identity()
7                  farm = str(current_user['farm_id'])
8                  farm_id = ObjectId(farm)
9

```

```
10         start_date = datetime.datetime.strptime(request.args
11             .get('start_date'), '%Y-%m-%d') if request.args.get('start_date')
12             ) else datetime.datetime.strptime("2023-01-01", '%Y-%m-%d')
13
14         end_date = datetime.datetime.strptime(request.args.
15             get('end_date'), '%Y-%m-%d') + datetime.timedelta(days=1) if
16             request.args.get('end_date') else datetime.datetime.strptime(
17                 "2030-01-01", '%Y-%m-%d')
18
19         name = request.args.get('name') if request.args.get(
20             'name') else ""
21
22         pond_name = request.args.get('pond_name') if request
23             .args.get('pond_name') else ""
24
25
26         pipeline = [
27             {
28                 '$match': {
29                     'created_at': {
30                         '$gte': start_date,
31                         '$lte': end_date,
32                     }
33                 }
34             },
35             {
36                 '$match': {
37                     "farm_id": farm_id,
38                     'pond': {
39                         '$regex': pond_name,
40                         '$options': 'i'
41                     }
42                 }
43             },
44             {"$sort": {"fish_feed_id": 1}},
45             {'$lookup': {
46                 'from': 'feed_inventory',
47                 'let': {"fishfeedid": "$fish_feed_id"},
48                 'pipeline': [
49                     {'$match': {'$expr': {'$eq': ['$id', '$fishfeedid']}},
50                     {
51                         '$match': {
52                             'brand_name': {
53                                 '$regex': name,
54                                 '$options': 'i'
55                             }
56                         }
57                     }
58                 ]
59             }
60         }
```

```

46     } ,
47     { "$project": {
48         "_id": 1,
49         "id_int": 1,
50         "feed_category": 1,
51         "brand_name": 1,
52         "price": 1,
53         "amount": 1,
54         "created_at": 1,
55     } }
56   ],
57   'as': 'feed'
58 },
59 { "$addFields": {
60     "feed": { "$first": "$feed" },
61   },
62 }
63
64 testing = FeedUsed.objects.aggregate(pipeline)
65 temp = list(testing)
66 response = json.dumps({
67     'status': 'success',
68     'data': temp,
69 }, default=str)
70 return Response(response, mimetype="application/json
", status=200)
71 except Exception as e:
72     response = {"message": e}
73 response = json.dumps(response, default=str)
74 return Response(response, mimetype="application/json
", status=400)
75

```

- Menambahkan riwayat pemakaian pakan (HTTP Method - POST)

```

1 class FeedFishHistoryApi(Resource):
2     @jwt_required()
3     def post(self):
4         try:
5             current_user = get_jwt_identity()
6             farm = str(current_user['farm_id'])
7
8             req_pond = request.form.get('pond', None)
9             req_feed_id = request.form.get('fish_feed_id', None)

```

```

10         req_usage = request.form.get('usage', None)
11
12         history_by_pond = FeedUsed.objects(pond=req_pond,
13                                         fish_feed_id=req_feed_id).first()
14
15         theDate = request.form.get('created_at', None)
16
17         body = {
18             "farm_id": farm,
19             "fish_feed_id": request.form.get('fish_feed_id',
20                                             None),
21             "original_amount": request.form.get(
22                 'original_amount', None),
23             "usage": request.form.get('usage', None),
24             "pond": request.form.get('pond', None),
25         }
26
27         if theDate != '':
28             body['created_at'] = datetime.datetime.strptime(
29                 theDate, "%Y-%m-%dT%H:%M:%S.%f %z")
30
31         except Exception as e:
32             response = {"message": str(e)}
33             response = json.dumps(response, default=str)
34             return Response(response, mimetype="application/json",
35                             status=400)

```

(d) Controller HTTP pada Flutter

- Mengambil semua data riwayat pemakaian pakan (HTTP Method - GET)

```

1     Future getHistoryFeedData(bool isReversed, String
2     firstDate, String lastDate,
3     Function() doAfter) async {
4     feedHistoryList.value.data!.clear();
5     isLoadingHistory.value = true;
6
7     SharedPreferences prefs = await SharedPreferences.
8     getInstance();
9     String token = prefs.getString('token').toString();
10    var headers = {'Authorization': 'Bearer $token'};

```

```

10         final response = await http.get(
11             Uri.parse('${Urls.feedSch}?start_date=$firstDate&
12             end_date=$lastDate'),
13             headers: headers);
14
15         try {
16             if (response.statusCode == 200) {
17                 HistoryFeedModel res =
18                     HistoryFeedModel.fromJson(jsonDecode(response.body));
19
20                 if (isReversed) {
21                     var temp = res;
22                     feedHistoryList.value.data = temp.data!.reversed.
23                     toList();
24                 } else {
25                     var temp = res;
26                     feedHistoryList.value.data = temp.data!;
27                 }
28
29                 doAfter();
30             }
31             catch (e) {
32                 throw Exception(e);
33             }
34             isLoadingHistory.value = false;
35         }
36     }
37 
```

- Menambahkan riwayat pemakaian pakan (HTTP Method - POST)

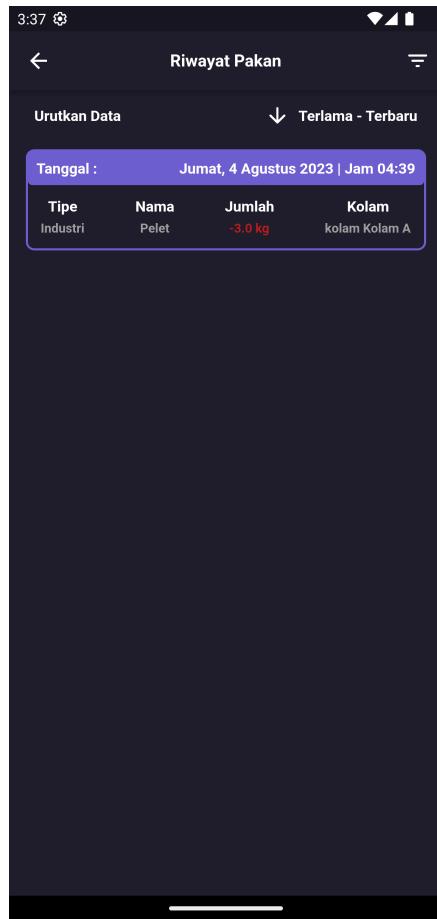
```

1     Future postHistoryFeedData(
2         String pondName,
3         String feedId,
4         String amount,
5         String used,
6         String usedDate,
7         Function() doAfter,
8     ) async {
9         var map = <String, dynamic>{};
10
11         SharedPreferences prefs = await SharedPreferences.
12             getInstance();
13         String token = prefs.getString('token').toString();
14 
```

```
13     var headers = {'Authorization': 'Bearer $token'};  
14  
15     map['pond'] = pondName;  
16     map['fish_feed_id'] = feedId;  
17     map['original_amount'] = amount;  
18     map['usage'] = used.replaceAll(',', '.');  
19     map['created_at'] = usedDate;  
20  
21     inspect(map);  
22  
23     try {  
24         final res = await http.post(  
25             Uri.parseUrls.feedSch),  
26             body: map,  
27             headers: headers,  
28         );  
29         if (res.statusCode != 200) {  
30             inspect(res);  
31         }  
32         doAfter();  
33     } catch (e) {  
34         throw Exception(e);  
35     }  
36 }  
37
```

(e) Tampilan pada Flutter

- Halaman Riwayat Pemakaian Pakan



Gambar 4.52: Halaman Riwayat Pemakaian Pakan

Pada halaman tersebut, dapat dilihat list dari pemakaian pakan yang terdiri dari tanggal, tipe pakan, nama pakan, jumlah dan kolam tempat pakan itu digunakan. Pada pojok kanan atas juga terdapat tombol filter untuk memfilter data list riwayat pakan sesuai dengan tanggal input.

6. Design route dan penerapan pada Flutter untuk riwayat pemakaian suplemen (dalam bentuk RESTful API)
 - (a) Design Sample Route

Berikut merupakan sample route yang sudah dibuat untuk riwayat pemakaian suplemen.

Group	Endpoint	HTTP Status	Operation	Purpose
History Inventaris Suplemen	/history/inventory/suplemen?start_date=&end_date=&pond_name=&narr	GET	READ	mengambil data riwayat penggunaan suplemen
	/history/inventory/suplemen	POST	CREATE	menginput data penggunaan suplemen

Gambar 4.53: Sample Route Riwayat Pemakaian Suplemen

(b) Model Backend

Berdasarkan sample route tersebut, dapat dibuat model dan class HTTP method pada backend. Berikut model serta class untuk riwayat pemakaian suplemen :

- Model Riwayat Pemakaian Suplemen

```

1      class SuplemenUsed(db.Document):
2          fish_suplemen_id = db.ReferenceField(SuplemenInventory,
3          required=True)
4          farm_id = db.ReferenceField(Farm, required=True)
5          original_amount = db.FloatField(required=True)
6          usage = db.FloatField(required=True)
7          pond = db.StringField(required=True)
8          created_at = db.DateTimeField(default=datetime.datetime.
now)
9          updated_at = db.DateTimeField(default=datetime.datetime.
now)

```

(c) Fungsi-fungsi HTTP Method

- Mengambil semua data riwayat pemakaian suplemen (HTTP Method - GET)

```

1      class SuplemenHistoryApi(Resource):
2          @jwt_required()
3
4          def get(self):
5              try:
6                  current_user = get_jwt_identity()
7                  farm = str(current_user['farm_id'])
8                  farm_id = ObjectId(farm)
9
10                 start_date = datetime.datetime.strptime(request.args
.get('start_date'), '%Y-%m-%d') if request.args.get('start_date'
) else datetime.datetime.strptime("2023-01-01", '%Y-%m-%d')

```

```

11         end_date = datetime.datetime.strptime(request.args.
12             get('end_date'), '%Y-%m-%d') + datetime.timedelta(days=1) if
13             request.args.get('end_date') else datetime.datetime.strptime(""
14                 "2030-01-01", '%Y-%m-%d')
15
16         name = request.args.get('name') if request.args.get(
17             'name') else ""
18
19         pond_name = request.args.get('pond_name') if request
20             .args.get('pond_name') else ""
21
22
23         pipeline = [
24             {
25                 '$match': {
26                     'created_at': {
27                         '$gte': start_date,
28                         '$lte': end_date,
29                     }
30                 }
31             },
32             {
33                 '$match': {
34                     "farm_id": farm_id,
35                     'pond': {
36                         '$regex': pond_name,
37                         '$options': 'i'
38                     }
39                 }
40             },
41             {"$sort": {"fish_suplemen_id": 1}},
42             {'$lookup': {
43                 'from': 'suplemen_inventory',
44                 'let': {"fishsuplemenid": "$fish_suplemen_id"},
45                 'pipeline': [
46                     {'$match': {'$expr': {'$eq': ['$id', '$fishsuplemenid']} }},
47                     {
48                         '$match': {
49                             'name': {
50                                 '$regex': name,
51                                 '$options': 'i'
52                             }
53                         }
54                     },
55                     {"$project": {
56                         "_id": 1,
57                         "name": 1,
58                         "farm_id": 1,
59                         "pond": 1,
60                         "suplemen_id": 1,
61                         "suplemen_name": 1,
62                         "suplemen_size": 1,
63                         "suplemen_weight": 1
64                     }}
65                 ]
66             }
67         ]
68     }
69
70     return pipeline
71
72
73     @app.route('/fish-suplemen/', methods=['PUT'])
74     def update_fish_suplemen(id):
75         suplemen = Suplemen.query.get(id)
76
77         if suplemen is None:
78             return jsonify({'error': 'Suplemen not found'}), 404
79
80         suplemen.suplemen_name = request.json['suplemen_name']
81         suplemen.suplemen_size = request.json['suplemen_size']
82         suplemen.suplemen_weight = request.json['suplemen_weight']
83
84         db.session.commit()
85
86         return jsonify(suplemen)
87
88
89     @app.route('/fish-suplemen', methods=['POST'])
90     def create_fish_suplemen():
91         suplemen = Suplemen(
92             suplemen_name=request.json['suplemen_name'],
93             suplemen_size=request.json['suplemen_size'],
94             suplemen_weight=request.json['suplemen_weight'])
95
96         db.session.add(suplemen)
97         db.session.commit()
98
99         return jsonify(suplemen)
100
101
102     @app.route('/fish-suplemen/', methods=['DELETE'])
103     def delete_fish_suplemen(id):
104         suplemen = Suplemen.query.get(id)
105
106         if suplemen is None:
107             return jsonify({'error': 'Suplemen not found'}), 404
108
109         db.session.delete(suplemen)
110         db.session.commit()
111
112         return jsonify({'message': 'Suplemen deleted successfully'})
113
114
115     @app.route('/fish-suplemen', methods=['GET'])
116     def get_fish_suplemen():
117         suplemen = Suplemen.query.all()
118
119         if suplemen is None:
120             return jsonify({'error': 'Suplemen not found'}), 404
121
122         return jsonify([suplemen])
123
124
125     @app.route('/fish-suplemen/', methods=['GET'])
126     def get_fish_suplemen_by_id(id):
127         suplemen = Suplemen.query.get(id)
128
129         if suplemen is None:
130             return jsonify({'error': 'Suplemen not found'}), 404
131
132         return jsonify(suplemen)
133
134
135     @app.route('/fish-suplemen/search', methods=['GET'])
136     def search_fish_suplemen():
137         start_date = request.args.get('start_date')
138         end_date = request.args.get('end_date')
139         name = request.args.get('name')
140         pond_name = request.args.get('pond_name')
141
142         suplemen = Suplemen.query
143
144         if start_date and end_date:
145             suplemen = suplemen.filter(
146                 Suplemen.created_at.between(start_date, end_date))
147
148         if name:
149             suplemen = suplemen.filter(Suplemen.name.ilike(f'%{name}%'))
150
151         if pond_name:
152             suplemen = suplemen.filter(Suplemen.pond.ilike(f'%{pond_name}%'))
153
154         suplemen = suplemen.order_by(Suplemen.id.desc())
155
156         return jsonify([suplemen])
157
158
159     @app.route('/fish-suplemen/filter', methods=['GET'])
160     def filter_fish_suplemen():
161         start_date = request.args.get('start_date')
162         end_date = request.args.get('end_date')
163         farm_id = request.args.get('farm_id')
164         pond_name = request.args.get('pond_name')
165
166         suplemen = Suplemen.query
167
168         if start_date and end_date:
169             suplemen = suplemen.filter(
170                 Suplemen.created_at.between(start_date, end_date))
171
172         if farm_id:
173             suplemen = suplemen.filter(Suplemen.farm_id == farm_id)
174
175         if pond_name:
176             suplemen = suplemen.filter(Suplemen.pond.ilike(f'%{pond_name}%'))
177
178         suplemen = suplemen.order_by(Suplemen.id.desc())
179
180         return jsonify([suplemen])
181
182
183     @app.route('/fish-suplemen/filter/lookup', methods=['GET'])
184     def filter_lookup_fish_suplemen():
185         start_date = request.args.get('start_date')
186         end_date = request.args.get('end_date')
187         farm_id = request.args.get('farm_id')
188         pond_name = request.args.get('pond_name')
189
190         suplemen = Suplemen.query
191
192         if start_date and end_date:
193             suplemen = suplemen.filter(
194                 Suplemen.created_at.between(start_date, end_date))
195
196         if farm_id:
197             suplemen = suplemen.filter(Suplemen.farm_id == farm_id)
198
199         if pond_name:
200             suplemen = suplemen.filter(Suplemen.pond.ilike(f'%{pond_name}%'))
201
202         suplemen = suplemen.order_by(Suplemen.id.desc())
203
204         return jsonify([suplemen])
205
206
207     @app.route('/fish-suplemen/filter/lookup/aggregate', methods=['GET'])
208     def filter_lookup_aggregate_fish_suplemen():
209         start_date = request.args.get('start_date')
210         end_date = request.args.get('end_date')
211         farm_id = request.args.get('farm_id')
212         pond_name = request.args.get('pond_name')
213
214         suplemen = Suplemen.query
215
216         if start_date and end_date:
217             suplemen = suplemen.filter(
218                 Suplemen.created_at.between(start_date, end_date))
219
220         if farm_id:
221             suplemen = suplemen.filter(Suplemen.farm_id == farm_id)
222
223         if pond_name:
224             suplemen = suplemen.filter(Suplemen.pond.ilike(f'%{pond_name}%'))
225
226         suplemen = suplemen.order_by(Suplemen.id.desc())
227
228         pipeline = [
229             {"$match": {"$expr": {"$eq": ["$id", "$fishsuplemenid"]}}},
230             {
231                 '$match': {
232                     'name': {
233                         '$regex': name,
234                         '$options': 'i'
235                     }
236                 }
237             },
238             {"$project": {
239                 "_id": 1,
240                 "name": 1,
241                 "farm_id": 1,
242                 "pond": 1,
243                 "suplemen_id": 1,
244                 "suplemen_name": 1,
245                 "suplemen_size": 1,
246                 "suplemen_weight": 1
247             }}
248         ]
249
250         suplemen = suplemen.pipeline(pipeline)
251
252         return jsonify([suplemen])
253
254
255     @app.route('/fish-suplemen/filter/lookup/aggregate/lookup', methods=['GET'])
256     def filter_lookup_aggregate_lookup_fish_suplemen():
257         start_date = request.args.get('start_date')
258         end_date = request.args.get('end_date')
259         farm_id = request.args.get('farm_id')
260         pond_name = request.args.get('pond_name')
261
262         suplemen = Suplemen.query
263
264         if start_date and end_date:
265             suplemen = suplemen.filter(
266                 Suplemen.created_at.between(start_date, end_date))
267
268         if farm_id:
269             suplemen = suplemen.filter(Suplemen.farm_id == farm_id)
270
271         if pond_name:
272             suplemen = suplemen.filter(Suplemen.pond.ilike(f'%{pond_name}%'))
273
274         suplemen = suplemen.order_by(Suplemen.id.desc())
275
276         pipeline = [
277             {"$match": {"$expr": {"$eq": ["$id", "$fishsuplemenid"]}}},
278             {
279                 '$match': {
280                     'name': {
281                         '$regex': name,
282                         '$options': 'i'
283                     }
284                 }
285             },
286             {"$lookup": {
287                 'from': 'suplemen_inventory',
288                 'let': {"fishsuplemenid": "$fish_suplemen_id"},
289                 'pipeline': [
290                     {'$match': {'$expr': {'$eq': ['$id', '$fishsuplemenid']}}}
291                 ]
292             }
293         ]
294
295         suplemen = suplemen.pipeline(pipeline)
296
297         return jsonify([suplemen])
298
299
300     @app.route('/fish-suplemen/filter/lookup/aggregate/lookup/lookup', methods=['GET'])
301     def filter_lookup_aggregate_lookup_lookup_fish_suplemen():
302         start_date = request.args.get('start_date')
303         end_date = request.args.get('end_date')
304         farm_id = request.args.get('farm_id')
305         pond_name = request.args.get('pond_name')
306
307         suplemen = Suplemen.query
308
309         if start_date and end_date:
310             suplemen = suplemen.filter(
311                 Suplemen.created_at.between(start_date, end_date))
312
313         if farm_id:
314             suplemen = suplemen.filter(Suplemen.farm_id == farm_id)
315
316         if pond_name:
317             suplemen = suplemen.filter(Suplemen.pond.ilike(f'%{pond_name}%'))
318
319         suplemen = suplemen.order_by(Suplemen.id.desc())
320
321         pipeline = [
322             {"$match": {"$expr": {"$eq": ["$id", "$fishsuplemenid"]}}},
323             {
324                 '$match': {
325                     'name': {
326                         '$regex': name,
327                         '$options': 'i'
328                     }
329                 }
330             },
331             {"$lookup": {
332                 'from': 'suplemen_inventory',
333                 'let': {"fishsuplemenid": "$fish_suplemen_id"},
334                 'pipeline': [
335                     {'$match': {'$expr': {'$eq': ['$id', '$fishsuplemenid']}}}
336                 ]
337             }
338         ]
339
340         suplemen = suplemen.pipeline(pipeline)
341
342         return jsonify([suplemen])
343
344
345     @app.route('/fish-suplemen/filter/lookup/aggregate/lookup/lookup/lookup', methods=['GET'])
346     def filter_lookup_aggregate_lookup_lookup_lookup_fish_suplemen():
347         start_date = request.args.get('start_date')
348         end_date = request.args.get('end_date')
349         farm_id = request.args.get('farm_id')
350         pond_name = request.args.get('pond_name')
351
352         suplemen = Suplemen.query
353
354         if start_date and end_date:
355             suplemen = suplemen.filter(
356                 Suplemen.created_at.between(start_date, end_date))
357
358         if farm_id:
359             suplemen = suplemen.filter(Suplemen.farm_id == farm_id)
360
361         if pond_name:
362             suplemen = suplemen.filter(Suplemen.pond.ilike(f'%{pond_name}%'))
363
364         suplemen = suplemen.order_by(Suplemen.id.desc())
365
366         pipeline = [
367             {"$match": {"$expr": {"$eq": ["$id", "$fishsuplemenid"]}}},
368             {
369                 '$match': {
370                     'name': {
371                         '$regex': name,
372                         '$options': 'i'
373                     }
374                 }
375             },
376             {"$lookup": {
377                 'from': 'suplemen_inventory',
378                 'let': {"fishsuplemenid": "$fish_suplemen_id"},
379                 'pipeline': [
380                     {'$match': {'$expr': {'$eq': ['$id', '$fishsuplemenid']}}}
381                 ]
382             }
383         ]
384
385         suplemen = suplemen.pipeline(pipeline)
386
387         return jsonify([suplemen])
388
389
390     @app.route('/fish-suplemen/filter/lookup/aggregate/lookup/lookup/lookup/lookup', methods=['GET'])
391     def filter_lookup_aggregate_lookup_lookup_lookup_lookup_fish_suplemen():
392         start_date = request.args.get('start_date')
393         end_date = request.args.get('end_date')
394         farm_id = request.args.get('farm_id')
395         pond_name = request.args.get('pond_name')
396
397         suplemen = Suplemen.query
398
399         if start_date and end_date:
400             suplemen = suplemen.filter(
401                 Suplemen.created_at.between(start_date, end_date))
402
403         if farm_id:
404             suplemen = suplemen.filter(Suplemen.farm_id == farm_id)
405
406         if pond_name:
407             suplemen = suplemen.filter(Suplemen.pond.ilike(f'%{pond_name}%'))
408
409         suplemen = suplemen.order_by(Suplemen.id.desc())
410
411         pipeline = [
412             {"$match": {"$expr": {"$eq": ["$id", "$fishsuplemenid"]}}},
413             {
414                 '$match': {
415                     'name': {
416                         '$regex': name,
417                         '$options': 'i'
418                     }
419                 }
420             },
421             {"$lookup": {
422                 'from': 'suplemen_inventory',
423                 'let': {"fishsuplemenid": "$fish_suplemen_id"},
424                 'pipeline': [
425                     {'$match': {'$expr': {'$eq': ['$id', '$fishsuplemenid']}}}
426                 ]
427             }
428         ]
429
430         suplemen = suplemen.pipeline(pipeline)
431
432         return jsonify([suplemen])
433
434
435     @app.route('/fish-suplemen/filter/lookup/aggregate/lookup/lookup/lookup/lookup/lookup', methods=['GET'])
436     def filter_lookup_aggregate_lookup_lookup_lookup_lookup_lookup_fish_suplemen():
437         start_date = request.args.get('start_date')
438         end_date = request.args.get('end_date')
439         farm_id = request.args.get('farm_id')
440         pond_name = request.args.get('pond_name')
441
442         suplemen = Suplemen.query
443
444         if start_date and end_date:
445             suplemen = suplemen.filter(
446                 Suplemen.created_at.between(start_date, end_date))
447
448         if farm_id:
449             suplemen = suplemen.filter(Suplemen.farm_id == farm_id)
450
451         if pond_name:
452             suplemen = suplemen.filter(Suplemen.pond.ilike(f'%{pond_name}%'))
453
454         suplemen = suplemen.order_by(Suplemen.id.desc())
455
456         pipeline = [
457             {"$match": {"$expr": {"$eq": ["$id", "$fishsuplemenid"]}}},
458             {
459                 '$match': {
460                     'name': {
461                         '$regex': name,
462                         '$options': 'i'
463                     }
464                 }
465             },
466             {"$lookup": {
467                 'from': 'suplemen_inventory',
468                 'let': {"fishsuplemenid": "$fish_suplemen_id"},
469                 'pipeline': [
470                     {'$match': {'$expr': {'$eq': ['$id', '$fishsuplemenid']}}}
471                 ]
472             }
473         ]
474
475         suplemen = suplemen.pipeline(pipeline)
476
477         return jsonify([suplemen])
478
479
480     @app.route('/fish-suplemen/filter/lookup/aggregate/lookup/lookup/lookup/lookup/lookup/lookup', methods=['GET'])
481     def filter_lookup_aggregate_lookup_lookup_lookup_lookup_lookup_lookup_fish_suplemen():
482         start_date = request.args.get('start_date')
483         end_date = request.args.get('end_date')
484         farm_id = request.args.get('farm_id')
485         pond_name = request.args.get('pond_name')
486
487         suplemen = Suplemen.query
488
489         if start_date and end_date:
490             suplemen = suplemen.filter(
491                 Suplemen.created_at.between(start_date, end_date))
492
493         if farm_id:
494             suplemen = suplemen.filter(Suplemen.farm_id == farm_id)
495
496         if pond_name:
497             suplemen = suplemen.filter(Suplemen.pond.ilike(f'%{pond_name}%'))
498
499         suplemen = suplemen.order_by(Suplemen.id.desc())
500
501         pipeline = [
502             {"$match": {"$expr": {"$eq": ["$id", "$fishsuplemenid"]}}},
503             {
504                 '$match': {
505                     'name': {
506                         '$regex': name,
507                         '$options': 'i'
508                     }
509                 }
510             },
511             {"$lookup": {
512                 'from': 'suplemen_inventory',
513                 'let': {"fishsuplemenid": "$fish_suplemen_id"},
514                 'pipeline': [
515                     {'$match': {'$expr': {'$eq': ['$id', '$fishsuplemenid']}}}
516                 ]
517             }
518         ]
519
520         suplemen = suplemen.pipeline(pipeline)
521
522         return jsonify([suplemen])
523
524
525     @app.route('/fish-suplemen/filter/lookup/aggregate/lookup/lookup/lookup/lookup/lookup/lookup/lookup', methods=['GET'])
526     def filter_lookup_aggregate_lookup_lookup_lookup_lookup_lookup_lookup_lookup_fish_suplemen():
527         start_date = request.args.get('start_date')
528         end_date = request.args.get('end_date')
529         farm_id = request.args.get('farm_id')
530         pond_name = request.args.get('pond_name')
531
532         suplemen = Suplemen.query
533
534         if start_date and end_date:
535             suplemen = suplemen.filter(
536                 Suplemen.created_at.between(start_date, end_date))
537
538         if farm_id:
539             suplemen = suplemen.filter(Suplemen.farm_id == farm_id)
540
541         if pond_name:
542             suplemen = suplemen.filter(Suplemen.pond.ilike(f'%{pond_name}%'))
543
544         suplemen = suplemen.order_by(Suplemen.id.desc())
545
546         pipeline = [
547             {"$match": {"$expr": {"$eq": ["$id", "$fishsuplemenid"]}}},
548             {
549                 '$match': {
550                     'name': {
551                         '$regex': name,
552                         '$options': 'i'
553                     }
554                 }
555             },
556             {"$lookup": {
557                 'from': 'suplemen_inventory',
558                 'let': {"fishsuplemenid": "$fish_suplemen_id"},
559                 'pipeline': [
560                     {'$match': {'$expr': {'$eq': ['$id', '$fishsuplemenid']}}}
561                 ]
562             }
563         ]
564
565         suplemen = suplemen.pipeline(pipeline)
566
567         return jsonify([suplemen])
568
569
570     @app.route('/fish-suplemen/filter/lookup/aggregate/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup', methods=['GET'])
571     def filter_lookup_aggregate_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_fish_suplemen():
572         start_date = request.args.get('start_date')
573         end_date = request.args.get('end_date')
574         farm_id = request.args.get('farm_id')
575         pond_name = request.args.get('pond_name')
576
577         suplemen = Suplemen.query
578
579         if start_date and end_date:
580             suplemen = suplemen.filter(
581                 Suplemen.created_at.between(start_date, end_date))
582
583         if farm_id:
584             suplemen = suplemen.filter(Suplemen.farm_id == farm_id)
585
586         if pond_name:
587             suplemen = suplemen.filter(Suplemen.pond.ilike(f'%{pond_name}%'))
588
589         suplemen = suplemen.order_by(Suplemen.id.desc())
590
591         pipeline = [
592             {"$match": {"$expr": {"$eq": ["$id", "$fishsuplemenid"]}}},
593             {
594                 '$match': {
595                     'name': {
596                         '$regex': name,
597                         '$options': 'i'
598                     }
599                 }
600             },
601             {"$lookup": {
602                 'from': 'suplemen_inventory',
603                 'let': {"fishsuplemenid": "$fish_suplemen_id"},
604                 'pipeline': [
605                     {'$match': {'$expr': {'$eq': ['$id', '$fishsuplemenid']}}}
606                 ]
607             }
608         ]
609
610         suplemen = suplemen.pipeline(pipeline)
611
612         return jsonify([suplemen])
613
614
615     @app.route('/fish-suplemen/filter/lookup/aggregate/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup', methods=['GET'])
616     def filter_lookup_aggregate_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_fish_suplemen():
617         start_date = request.args.get('start_date')
618         end_date = request.args.get('end_date')
619         farm_id = request.args.get('farm_id')
620         pond_name = request.args.get('pond_name')
621
622         suplemen = Suplemen.query
623
624         if start_date and end_date:
625             suplemen = suplemen.filter(
626                 Suplemen.created_at.between(start_date, end_date))
627
628         if farm_id:
629             suplemen = suplemen.filter(Suplemen.farm_id == farm_id)
630
631         if pond_name:
632             suplemen = suplemen.filter(Suplemen.pond.ilike(f'%{pond_name}%'))
633
634         suplemen = suplemen.order_by(Suplemen.id.desc())
635
636         pipeline = [
637             {"$match": {"$expr": {"$eq": ["$id", "$fishsuplemenid"]}}},
638             {
639                 '$match': {
640                     'name': {
641                         '$regex': name,
642                         '$options': 'i'
643                     }
644                 }
645             },
646             {"$lookup": {
647                 'from': 'suplemen_inventory',
648                 'let': {"fishsuplemenid": "$fish_suplemen_id"},
649                 'pipeline': [
650                     {'$match': {'$expr': {'$eq': ['$id', '$fishsuplemenid']}}}
651                 ]
652             }
653         ]
654
655         suplemen = suplemen.pipeline(pipeline)
656
657         return jsonify([suplemen])
658
659
660     @app.route('/fish-suplemen/filter/lookup/aggregate/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup', methods=['GET'])
661     def filter_lookup_aggregate_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_fish_suplemen():
662         start_date = request.args.get('start_date')
663         end_date = request.args.get('end_date')
664         farm_id = request.args.get('farm_id')
665         pond_name = request.args.get('pond_name')
666
667         suplemen = Suplemen.query
668
669         if start_date and end_date:
670             suplemen = suplemen.filter(
671                 Suplemen.created_at.between(start_date, end_date))
672
673         if farm_id:
674             suplemen = suplemen.filter(Suplemen.farm_id == farm_id)
675
676         if pond_name:
677             suplemen = suplemen.filter(Suplemen.pond.ilike(f'%{pond_name}%'))
678
679         suplemen = suplemen.order_by(Suplemen.id.desc())
680
681         pipeline = [
682             {"$match": {"$expr": {"$eq": ["$id", "$fishsuplemenid"]}}},
683             {
684                 '$match': {
685                     'name': {
686                         '$regex': name,
687                         '$options': 'i'
688                     }
689                 }
690             },
691             {"$lookup": {
692                 'from': 'suplemen_inventory',
693                 'let': {"fishsuplemenid": "$fish_suplemen_id"},
694                 'pipeline': [
695                     {'$match': {'$expr': {'$eq': ['$id', '$fishsuplemenid']}}}
696                 ]
697             }
698         ]
699
700         suplemen = suplemen.pipeline(pipeline)
701
702         return jsonify([suplemen])
703
704
705     @app.route('/fish-suplemen/filter/lookup/aggregate/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup', methods=['GET'])
706     def filter_lookup_aggregate_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_fish_suplemen():
707         start_date = request.args.get('start_date')
708         end_date = request.args.get('end_date')
709         farm_id = request.args.get('farm_id')
710         pond_name = request.args.get('pond_name')
711
712         suplemen = Suplemen.query
713
714         if start_date and end_date:
715             suplemen = suplemen.filter(
716                 Suplemen.created_at.between(start_date, end_date))
717
718         if farm_id:
719             suplemen = suplemen.filter(Suplemen.farm_id == farm_id)
720
721         if pond_name:
722             suplemen = suplemen.filter(Suplemen.pond.ilike(f'%{pond_name}%'))
723
724         suplemen = suplemen.order_by(Suplemen.id.desc())
725
726         pipeline = [
727             {"$match": {"$expr": {"$eq": ["$id", "$fishsuplemenid"]}}},
728             {
729                 '$match': {
730                     'name': {
731                         '$regex': name,
732                         '$options': 'i'
733                     }
734                 }
735             },
736             {"$lookup": {
737                 'from': 'suplemen_inventory',
738                 'let': {"fishsuplemenid": "$fish_suplemen_id"},
739                 'pipeline': [
740                     {'$match': {'$expr': {'$eq': ['$id', '$fishsuplemenid']}}}
741                 ]
742             }
743         ]
744
745         suplemen = suplemen.pipeline(pipeline)
746
747         return jsonify([suplemen])
748
749
750     @app.route('/fish-suplemen/filter/lookup/aggregate/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup', methods=['GET'])
751     def filter_lookup_aggregate_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_fish_suplemen():
752         start_date = request.args.get('start_date')
753         end_date = request.args.get('end_date')
754         farm_id = request.args.get('farm_id')
755         pond_name = request.args.get('pond_name')
756
757         suplemen = Suplemen.query
758
759         if start_date and end_date:
760             suplemen = suplemen.filter(
761                 Suplemen.created_at.between(start_date, end_date))
762
763         if farm_id:
764             suplemen = suplemen.filter(Suplemen.farm_id == farm_id)
765
766         if pond_name:
767             suplemen = suplemen.filter(Suplemen.pond.ilike(f'%{pond_name}%'))
768
769         suplemen = suplemen.order_by(Suplemen.id.desc())
770
771         pipeline = [
772             {"$match": {"$expr": {"$eq": ["$id", "$fishsuplemenid"]}}},
773             {
774                 '$match': {
775                     'name': {
776                         '$regex': name,
777                         '$options': 'i'
778                     }
779                 }
780             },
781             {"$lookup": {
782                 'from': 'suplemen_inventory',
783                 'let': {"fishsuplemenid": "$fish_suplemen_id"},
784                 'pipeline': [
785                     {'$match': {'$expr': {'$eq': ['$id', '$fishsuplemenid']}}}
786                 ]
787             }
788         ]
789
790         suplemen = suplemen.pipeline(pipeline)
791
792         return jsonify([suplemen])
793
794
795     @app.route('/fish-suplemen/filter/lookup/aggregate/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup', methods=['GET'])
796     def filter_lookup_aggregate_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_fish_suplemen():
797         start_date = request.args.get('start_date')
798         end_date = request.args.get('end_date')
799         farm_id = request.args.get('farm_id')
800         pond_name = request.args.get('pond_name')
801
802         suplemen = Suplemen.query
803
804         if start_date and end_date:
805             suplemen = suplemen.filter(
806                 Suplemen.created_at.between(start_date, end_date))
807
808         if farm_id:
809             suplemen = suplemen.filter(Suplemen.farm_id == farm_id)
810
811         if pond_name:
812             suplemen = suplemen.filter(Suplemen.pond.ilike(f'%{pond_name}%'))
813
814         suplemen = suplemen.order_by(Suplemen.id.desc())
815
816         pipeline = [
817             {"$match": {"$expr": {"$eq": ["$id", "$fishsuplemenid"]}}},
818             {
819                 '$match': {
820                     'name': {
821                         '$regex': name,
822                         '$options': 'i'
823                     }
824                 }
825             },
826             {"$lookup": {
827                 'from': 'suplemen_inventory',
828                 'let': {"fishsuplemenid": "$fish_suplemen_id"},
829                 'pipeline': [
830                     {'$match': {'$expr': {'$eq': ['$id', '$fishsuplemenid']}}}
831                 ]
832             }
833         ]
834
835         suplemen = suplemen.pipeline(pipeline)
836
837         return jsonify([suplemen])
838
839
840     @app.route('/fish-suplemen/filter/lookup/aggregate/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup', methods=['GET'])
841     def filter_lookup_aggregate_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_fish_suplemen():
842         start_date = request.args.get('start_date')
843         end_date = request.args.get('end_date')
844         farm_id = request.args.get('farm_id')
845         pond_name = request.args.get('pond_name')
846
847         suplemen = Suplemen.query
848
849         if start_date and end_date:
850             suplemen = suplemen.filter(
851                 Suplemen.created_at.between(start_date, end_date))
852
853         if farm_id:
854             suplemen = suplemen.filter(Suplemen.farm_id == farm_id)
855
856         if pond_name:
857             suplemen = suplemen.filter(Suplemen.pond.ilike(f'%{pond_name}%'))
858
859         suplemen = suplemen.order_by(Suplemen.id.desc())
860
861         pipeline = [
862             {"$match": {"$expr": {"$eq": ["$id", "$fishsuplemenid"]}}},
863             {
864                 '$match': {
865                     'name': {
866                         '$regex': name,
867                         '$options': 'i'
868                     }
869                 }
870             },
871             {"$lookup": {
872                 'from': 'suplemen_inventory',
873                 'let': {"fishsuplemenid": "$fish_suplemen_id"},
874                 'pipeline': [
875                     {'$match': {'$expr': {'$eq': ['$id', '$fishsuplemenid']}}}
876                 ]
877             }
878         ]
879
880         suplemen = suplemen.pipeline(pipeline)
881
882         return jsonify([suplemen])
883
884
885     @app.route('/fish-suplemen/filter/lookup/aggregate/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup', methods=['GET'])
886     def filter_lookup_aggregate_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_fish_suplemen():
887         start_date = request.args.get('start_date')
888         end_date = request.args.get('end_date')
889         farm_id = request.args.get('farm_id')
890         pond_name = request.args.get('pond_name')
891
892         suplemen = Suplemen.query
893
894         if start_date and end_date:
895             suplemen = suplemen.filter(
896                 Suplemen.created_at.between(start_date, end_date))
897
898         if farm_id:
899             suplemen = suplemen.filter(Suplemen.farm_id == farm_id)
900
901         if pond_name:
902             suplemen = suplemen.filter(Suplemen.pond.ilike(f'%{pond_name}%'))
903
904         suplemen = suplemen.order_by(Suplemen.id.desc())
905
906         pipeline = [
907             {"$match": {"$expr": {"$eq": ["$id", "$fishsuplemenid"]}}},
908             {
909                 '$match': {
910                     'name': {
911                         '$regex': name,
912                         '$options': 'i'
913                     }
914                 }
915             },
916             {"$lookup": {
917                 'from': 'suplemen_inventory',
918                 'let': {"fishsuplemenid": "$fish_suplemen_id"},
919                 'pipeline': [
920                     {'$match': {'$expr': {'$eq': ['$id', '$fishsuplemenid']}}}
921                 ]
922             }
923         ]
924
925         suplemen = suplemen.pipeline(pipeline)
926
927         return jsonify([suplemen])
928
929
930     @app.route('/fish-suplemen/filter/lookup/aggregate/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup', methods=['GET'])
931     def filter_lookup_aggregate_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_fish_suplemen():
932         start_date = request.args.get('start_date')
933         end_date = request.args.get('end_date')
934         farm_id = request.args.get('farm_id')
935         pond_name = request.args.get('pond_name')
936
937         suplemen = Suplemen.query
938
939         if start_date and end_date:
940             suplemen = suplemen.filter(
941                 Suplemen.created_at.between(start_date, end_date))
942
943         if farm_id:
944             suplemen = suplemen.filter(Suplemen.farm_id == farm_id)
945
946         if pond_name:
947             suplemen = suplemen.filter(Suplemen.pond.ilike(f'%{pond_name}%'))
948
949         suplemen = suplemen.order_by(Suplemen.id.desc())
950
951         pipeline = [
952             {"$match": {"$expr": {"$eq": ["$id", "$fishsuplemenid"]}}},
953             {
954                 '$match': {
955                     'name': {
956                         '$regex': name,
957                         '$options': 'i'
958                     }
959                 }
960             },
961             {"$lookup": {
962                 'from': 'suplemen_inventory',
963                 'let': {"fishsuplemenid": "$fish_suplemen_id"},
964                 'pipeline': [
965                     {'$match': {'$expr': {'$eq': ['$id', '$fishsuplemenid']}}}
966                 ]
967             }
968         ]
969
970         suplemen = suplemen.pipeline(pipeline)
971
972         return jsonify([suplemen])
973
974
975     @app.route('/fish-suplemen/filter/lookup/aggregate/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup', methods=['GET'])
976     def filter_lookup_aggregate_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_fish_suplemen():
977         start_date = request.args.get('start_date')
978         end_date = request.args.get('end_date')
979         farm_id = request.args.get('farm_id')
980         pond_name = request.args.get('pond_name')
981
982         suplemen = Suplemen.query
983
984         if start_date and end_date:
985             suplemen = suplemen.filter(
986                 Suplemen.created_at.between(start_date, end_date))
987
988         if farm_id:
989             suplemen = suplemen.filter(Suplemen.farm_id == farm_id)
990
991         if pond_name:
992             suplemen = suplemen.filter(Suplemen.pond.ilike(f'%{pond_name}%'))
993
994         suplemen = suplemen.order_by(Suplemen.id.desc())
995
996         pipeline = [
997             {"$match": {"$expr": {"$eq": ["$id", "$fishsuplemenid"]}}},
998             {
999                 '$match': {
1000                     'name': {
1001                         '$regex': name,
1002                         '$options': 'i'
1003                     }
1004                 }
1005             },
1006             {"$lookup": {
1007                 'from': 'suplemen_inventory',
1008                 'let': {"fishsuplemenid": "$fish_suplemen_id"},
1009                 'pipeline': [
1010                     {'$match': {'$expr': {'$eq': ['$id', '$fishsuplemenid']}}}
1011                 ]
1012             }
1013         ]
1014
1015         suplemen = suplemen.pipeline(pipeline)
1016
1017         return jsonify([suplemen])
1018
1019
1020     @app.route('/fish-suplemen/filter/lookup/aggregate/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup/lookup', methods=['GET'])
1021     def filter_lookup_aggregate_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_lookup_fish_suplemen():
1022         start_date = request.args.get('start_date')
1023         end_date = request.args.get('end_date')
1024         farm_id = request.args.get('farm_id')
1025         pond_name = request.args.get('pond_name')
1026
1027         suplemen = Suplemen.query
1028
1029         if start_date and end_date:
1030             suplemen = suplemen.filter(
1031                 Suplemen.created_at.between(start_date, end_date))
1032
1033         if farm_id:
1034             suplemen = suplemen.filter(Suplemen.farm_id == farm_id)
1035
1036         if pond_name:
1037             suplemen = suplemen.filter(Suplemen.pond.ilike(f'%{pond_name}%'))
1038
1039         suplemen = suplemen.order_by(Suplemen.id.desc())
1040
1041         pipeline = [
1042             {"$match": {"$expr": {"$eq": ["$id", "$fishsuplemenid"]}}},
1043             {
1044                 '$match': {
1045                     'name': {
1046                         '$regex': name,
1047                         '$options': 'i'
1048                     }
1049                 }
1050             },
1051             {"$lookup": {
1052                 'from': 'suplemen_inventory',
1053                 'let': {"fishsuplemenid": "$fish_suplemen_id"},
1054                 'pipeline': [
1055                     {'$match': {'$expr': {'$eq': ['$id', '$fishsuplemenid']}}}
1056                 ]
1057             }
1
```

```

49         "id_int": 1,
50         "function": 1,
51         "name": 1,
52         "description": 1,
53         "price": 1,
54         "amount": 1,
55         "type": 1,
56         "min_expired_period": 1,
57         "max_expired_period": 1,
58         "image": 1,
59         "created_at": 1,
60     } }
61 ],
62 'as': 'suplemen'
63 } },
64 {"$addFields": {
65     "suplemen": {"$first": "$suplemen"},
66 } },
67 ]
68
69     testing = SuplemenUsed.objects.aggregate(pipeline)
70     temp = list(testing)
71     response = json.dumps({
72         'status': 'success',
73         'data': temp,
74     }, default=str)
75     return Response(response, mimetype="application/json
76     ", status=200)
77 except Exception as e:
78     response = {"message": e}
79     response = json.dumps(response, default=str)
80     return Response(response, mimetype="application/json
81     ", status=400)
82

```

- Menambahkan data riwayat pemakaian suplemen (HTTP Method - POST)

```

1     class SuplemenHistoryApi(Resource):
2         @jwt_required()
3         def post(self):
4             try:
5                 current_user = get_jwt_identity()
6                 farm = str(current_user['farm_id'])

```

```

7         req_pond = request.form.get('pond', None)
8         req_suplemen_id = request.form.get('fish_suplemen_id',
9             None)
10        req_usage = request.form.get('usage', None)
11
12        history_by_pond = SuplemenUsed.objects(pond=
13             req_pond, fish_suplemen_id=req_suplemen_id).first()
14
15        theDate = request.form.get('created_at', None)
16
17        body = {
18            "farm_id": farm,
19            "fish_suplemen_id": request.form.get(
20                'fish_suplemen_id', None),
21            "original_amount": request.form.get(
22                'original_amount', None),
23            "usage": request.form.get('usage', None),
24            "pond": request.form.get('pond', None),
25        }
26
27        if theDate != '':
28            body['created_at'] = datetime.datetime.strptime(
29                theDate, "%Y-%m-%dT%H:%M:%S.%f %z")
30
31        except Exception as e:
32            response = {"message": str(e)}
33            response = json.dumps(response, default=str)
34            return Response(response, mimetype="application/json",
35                            status=400)

```

(d) Controller HTTP pada Flutter

- Mengambil semua data riwayat pemakaian suplemen (HTTP Method - GET)

```
1         Future getHistorySuplemenData(bool isReversed, String
2             firstDate,
3
4             String lastDate, Function() doAfter) async {
5
6             suplemenHistoryList.value.data!.clear();
7
8             isLoadingHistory.value = true;
```

```
6     Sharedpreferences prefs = await Sharedpreferences.  
7     getInstance();  
8     String token = prefs.getString('token').toString();  
9     var headers = {'Authorization': 'Bearer $token'};  
10  
11     final response = await http.get(  
12         Uri.parse('${Urls.suplemenSch}?start_date=$firstDate&  
13         end_date=$lastDate'),  
14         headers: headers,  
15     );  
16  
17     try {  
18         if (response.statusCode == 200) {  
19             HistorySuplemenModel res =  
20                 HistorySuplemenModel.fromJson(jsonDecode(response.  
21                 body));  
22  
23             if (isReversed) {  
24                 var temp = res;  
25                 suplemenHistoryList.value.data = temp.data!.reversed.  
26                 toList();  
27  
28             } else {  
29                 var temp = res;  
30                 suplemenHistoryList.value.data = temp.data!;  
31             }  
32  
33             doAfter();  
34         }  
35     } catch (e) {  
36         throw Exception(e);  
37     }  
38     isLoadingHistory.value = false;  
39 }  
40
```

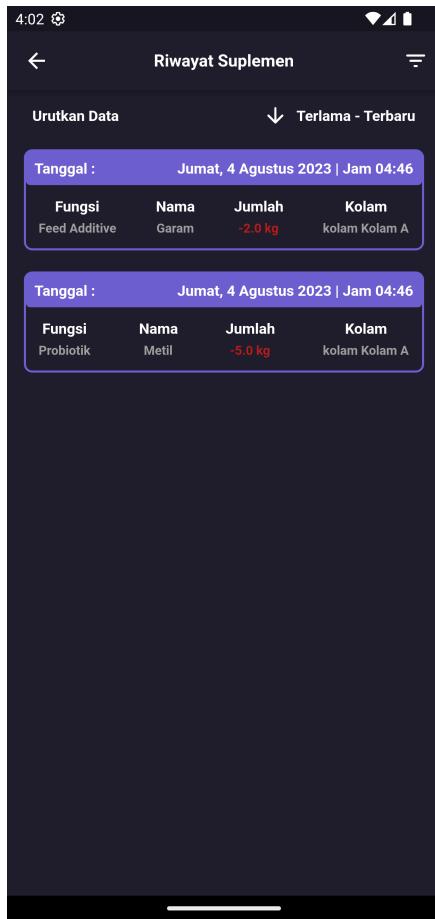
- Menambahkan data riwayat pemakaian suplemen (HTTP Method - POST)

```
1     Future postHistorySuplemenData(String pondName, List
2         suplemen,
3         String usedDate, Function() doAfter) async {
4             var map = <String, dynamic>{};
5
6             SharedPreferences prefs = await SharedPreferences.
```

```
5           getInstance();
6
7           String token = prefs.getString('token').toString();
8           var headers = {'Authorization': 'Bearer $token'};
9
10          map['pond'] = pondName;
11
12          for (var i = 0; i < suplemen.length; i++) {
13              map['fish_suplemen_id'] = suplemen[i]['suplemen_id'];
14              map['original_amount'] = suplemen[i]['original_value'];
15              map['usage'] = suplemen[i]['amount'];
16              map['created_at'] = usedDate;
17
18              inspect(map);
19
20              try {
21                  await http.post(
22                      Uri.parseUrls.suplemenSch),
23                      body: map,
24                      headers: headers,
25                      );
26                      doAfter();
27                  } catch (e) {
28                      throw Exception(e);
29                  }
30              }
31      }
```

(e) Tampilan pada Flutter

- Halaman Riwayat Pemakaian Suplemen



Gambar 4.54: Halaman Riwayat Pemakaian Suplemen

Pada halaman tersebut, dapat dilihat list dari pemakaian pakan yang terdiri dari tanggal, fungsi suplemen, nama suplemen, jumlah dan kolam tempat suplemen itu digunakan. Pada pojok kanan atas juga terdapat tombol filter untuk memfilter data list riwayat suplemen sesuai dengan tanggal input.

7. Integrasi data inventaris pakan pada halaman entry pakan

Dalam penggunaan fitur inventaris pakan, pakan yang sudah ada pada inventaris dapat digunakan pada halaman entry pakan. Pada bagian ini, dilakukan beberapa perubahan pada backend dan frontend untuk menyatukan antara inventaris pakan dengan riwayat pakan.

- Model backend riwayat pakan

```

1      class FeedHistory(db.Document):
2          pond_id = db.ReferenceField(Pond, required=True)
3          fish_feed_id = db.ReferenceField(FeedInventory, required=True)
4          farm_id = db.ReferenceField(Farm, required=True)
5          pond_activation_id = db.ReferenceField(PondActivation, required
6          =True)
7          feed_dose = db.FloatField(required=True)
8          feed_history_time = db.DateTimeField(default=datetime.datetime.
9          now)
10         created_at = db.DateTimeField(default=datetime.datetime.now)
11         updated_at = db.DateTimeField(default=datetime.datetime.now)
12

```

Pada tabel ini, terdapat feed_dose yang nantinya akan diisi dengan jumlah pakan yang digunakan serta fish_feed_id yang digunakan untuk memproyeksikan data pada inventaris pakan sehingga detail dari pakan tersebut dapat terlihat.

Berikut merupakan class dari pemberian pakan yang sudah diintegrasikan dengan inventaris pakan. Source code dapat dilihat sebagai berikut.

- Class pemberian pakan

```

1      class FeedHistorysApi(Resource):
2          @jwt_required()
3          def post(self):
4              try:
5                  pond_id = request.form.get("pond_id", None)
6                  print(pond_id)
7                  fish_feed_id = request.form.get("fish_feed_id", None)
8                  pond = Pond.objects.get(id=pond_id)
9                  if pond['isActive'] == False:
10                      response = {"message": "pond is not active"}
11                      response = json.dumps(response, default=str)
12                      return Response(response, mimetype="application/json",
13                      status=400)
13                      pond_activation = PondActivation.objects(
14                          pond_id=pond_id, isFinish=False).order_by('-activated_at'
15                          ).first()
15                      # feed_type = FeedType.objects.get(id=feed_type_id)

```

```

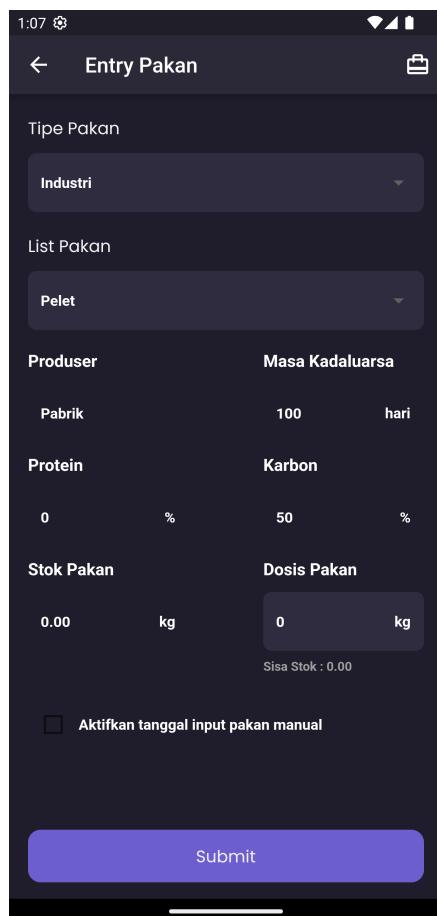
16         feed_history_time = request.form.get("feed_history_time",
17             None)
18         if feed_history_time != None:
19             feed_history_time = datetime.datetime.fromisoformat(
20                 feed_history_time)
21
22         current_user = get_jwt_identity()
23         farm = str(current_user['farm_id'])
24
25         body = {
26             "pond_id": pond_id,
27             "farm_id": farm,
28             "pond_activation_id": pond_activation.id,
29             "fish_feed_id": fish_feed_id,
30             "feed_dose": request.form.get("feed_dose", None),
31             "feed_history_time": feed_history_time
32         }
33
34         # # update feed inventory table
35         get_feed_by_id = FeedInventory.objects.get(id=request.form.
36             get('fish_feed_id', None))
37         get_feed_by_id.amount -= float(request.form.get('feed_dose'
38             , None))
39         get_feed_by_id.save()
40
41         feedhistory = FeedHistory(**body).save()
42         id = feedhistory.id
43         return {'id': str(id), 'message': 'success input'}, 200
44     except Exception as e:
45         response = {"message": str(e)}
46         response = json.dumps(response, default=str)
47         return Response(response, mimetype="application/json",
48                         status=400)
49
50

```

Fungsi pada class tersebut merupakan fungsi HTTP Method POST yang dimana fungsi itu berisi bagaimana proses pemberian pakan berlangsung. Nilai feed_dose pada body request tersebut nantinya akan digunakan menjadi substraksi dengan jumlah pakan yang ada di inventaris pakan. Kemudian, data tersebut nantinya akan dikirim ke tabel FeedHistory (tabel riwayat pemakaian

pakan).

Pada Flutter, halaman entry pakan yang sudah terintegrasi dengan inventaris pakan dapat dilihat sebagai berikut.



Gambar 4.55: Halaman Entry Pakan

Di halaman tersebut, terdapat beberapa detail informasi pakan serta form input dosis pakan yang diperlukan.

Adapun fungsi dari method POST pakan untuk mengirimkan data entry pakan kepada backend pada Flutter dapat dilihat pada code berikut.

```

1 Future<void> postFeedHistory() async {
2     bool value = await FeedHistoryService().postFeedHistory(
3         pondId: pond.id,
4         fishFeedId: pakanState.selectedFeedName.value['feed_id'],
5         feedDose: pakanState.amountChecker(feedDosisController.text),

```

```

6      doAfter: () async {
7          await feedController.getWeeklyRecapFeedHistory(
8              activation_id: activation.id.toString());
9          });
10
11         await pakanState.postHistoryFeedData(
12             pakanState.pondName.value,
13             pakanState.selectedFeedName.value['feed_id'],
14             pakanState.amount.text,
15             pakanState.amountChecker(feedDosisController.text),
16             pakanState.selectedUsedDate.value,
17             () => null,
18         );
19     }
20

```

Fungsi tersebut akan berjalan ketika tombol Submit pada halaman Entry Pakan ditekan. Pada fungsi tersebut, dijalankan dua jenis method POST yaitu method untuk mengirimkan data entry pakan ke backend dan mengirimkan penggunaan pakan ke backend yang nantinya akan masuk ke tabel riwayat penggunaan pakan.

8. Sprint 4 Review

Hasil review pada Sprint 4 ini adalah review dan testing oleh penulis selaku developer dengan Scrum Master. Setelah dilakukan testing, Scrum Master menyimpulkan bahwa penerapan fitur inventaris pakan, suplemen, listrik, aset serta riwayat pemakaian pakan dan suplemen telah berjalan dengan baik. Integrasi antara inventaris benih dengan aktivasi kolam serta integrasi antara inventaris pakan dengan entry pakan juga telah berjalan dengan baik.

5. Sprint 5

Sprint 5 dilaksanakan pada tanggal 12 Juli 2023 - 01 Agustus 2023. Detail dari Sprint 5 ini adalah mengerjakan tugas yang ada pada Sprint 5 Backlog di tabel berikut.

Tabel 4.5: Sprint 5 Backlog

No	Stories	Task	Status
1	Fitur pencatatan inventaris	- Design route dan penerapan pada Flutter untuk merk di inventaris pakan (dalam bentuk RESTful API)	Selesai
2	Fitur Treatment kolam yang terkoneksi dengan inventaris	- Integrasi halaman Input Treatment dengan data inventaris suplemen	Selesai
3	Fitur Panen termasuk harga nilai jual ikan	- Integrasi harga jual minimum ikan pada fitur Panen	Selesai
4	Fitur Pembukuan musim budidaya	- Design route dan penerapan Flutter untuk rekapitulasi panen beserta harga jual ikan (dalam bentuk RESTful API)	Selesai

Dalam proses Sprint ini, beberapa tahapan yang dilakukan adalah sebagai berikut :

1. Design route dan penerapan pada Flutter untuk merk di inventaris pakan (dalam bentuk RESTful API)

(a) Design Sample Route

Berikut merupakan sample route yang sudah dibuat untuk merk pakan.

Group	Endpoint	HTTP Status	Operation	Purpose
Merk Pakan	/inventory/feed/name	GET	READ	mengambil semua data merk pakan
	/inventory/feed/name/{id}	GET	READ	mengambil data spesifik dari merk pakan dengan id
	/inventory/feed/name	POST	CREATE	menginput data kedalam merk pakan
	/inventory/feed/name/{id}	PUT	UPDATE	memperbarui spesifik data pada merk pakan dengan id
	/inventory/feed/name/{id}	DELETE	DELETE	menghapus spesifik data pada merk pakan dengan id

Gambar 4.56: Sample Route Merk Pakan

2. Model Backend

Berdasarkan sample route tersebut, dapat dibuat model dan class HTTP method pada backend. Berikut model serta class untuk merk pakan :

```

1   class FeedName(db.Document):
2       id_int = db.SequenceField(required=True)
3       farm_id = db.ReferenceField(Farm, required=True)
4       type = db.StringField(required=True)
5       name = db.StringField(required=True)
6       description = db.StringField(required=True)
7       producer = db.StringField(required=True)
8       protein = db.IntField(required=True)
9       carbohydrate = db.IntField(required=True)
10      min_expired_period = db.IntField(required=True)
11      max_expired_period = db.IntField(required=True)
12      image = db.StringField(required=True)
13      created_at = db.DateTimeField(default=datetime.datetime.now)
14      updated_at = db.DateTimeField(default=datetime.datetime.now)
15

```

3. Fungsi-fungsi HTTP Method

- Mengambil semua data merk pakan (HTTP Method - GET)

```

1   class FeedNamesApi(Resource):
2       @jwt_required()
3       def get(self):
4           try:
5               current_user = get_jwt_identity()
6               farm = str(current_user['farm_id'])
7               farm_id = ObjectId(farm)
8
9               type = request.args.get('type') if request.args.get('type')
else ""
10
11               pipeline = [
12                   {"$sort": {"id_int": 1}},
13                   {
14                       '$match': {
15                           "farm_id": farm_id,
16                           'type': {
17                               '$regex': type,
18                               '$options': 'i'
19

```

```

19         }
20     }
21   }
22 ]
23
24   testing = FeedName.objects.aggregate(pipeline)
25   temp = list(testing)
26   response = json.dumps({
27     'status': 'success',
28     'data': temp,
29   }, default=str)
30   return Response(response, mimetype="application/json",
31   status=200)
32 except Exception as e:
33   response = {"message": e}
34   response = json.dumps(response, default=str)
35   return Response(response, mimetype="application/json",
36   status=400)
37

```

- Mengambil spesifik data merk pakan (HTTP Method - GET)

```

1   class FeedNameApi(Resource):
2       def get(self, id):
3           try:
4               pipeline = [
5                   {"$match": {"id_int": int(id)}},
6               ]
7
8               testing = FeedName.objects.aggregate(pipeline)
9               temp = list(testing)
10              if len(temp) == 0:
11                  res = {"message": 'no data found'}
12                  response = json.dumps(res, default=str)
13                  return Response(response, mimetype="application/json",
14                  status=200)
15                  response = json.dumps({
16                    'status': 'success',
17                    'data': temp[0],
18                  }, default=str)
19                  return Response(response, mimetype="application/json",
20                  status=200)
21
22 except Exception as e:
23     response = {"message": e}

```

```

21         response = json.dumps(response, default=str)
22
23     return Response(response, mimetype="application/json",
24                     status=400)

```

- Menambahkan data merk pakan (HTTP Method - POST)

```

1      class FeedNamesApi(Resource):
2          @jwt_required()
3
4          def post(self):
5              try:
6                  current_user = get_jwt_identity()
7                  farm = str(current_user['farm_id'])
8                  body = {
9                      "farm_id": farm,
10                     "type": request.form.get('type', None),
11                     "name": request.form.get('name', None),
12                     "description": request.form.get('description', None),
13                     "producer": request.form.get('producer', None),
14                     "protein": request.form.get('protein', None),
15                     "carbohydrate": request.form.get('carbohydrate', None),
16                     "min_expired_period": request.form.get(
17                         'min_expired_period', None),
18                     "max_expired_period": request.form.get(
19                         'max_expired_period', None),
20                         "image": request.form.get('image', None),
21                     }
22
23             feed_name = FeedName(**body).save()
24             id = feed_name.id
25             res = {"message": "success add feed name to db", "id": id,
26 "data": body}
27
28             response = json.dumps(res, default=str)
29             return Response(response, mimetype="application/json",
30                             status=200)
31
32         except Exception as e:
33             response = {"message": str(e)}
34             response = json.dumps(response, default=str)
35             return Response(response, mimetype="application/json",
36                             status=400)

```

- Mengambil semua data merk pakan (HTTP Method - PUT)

```

1      class FeedNameApi(Resource):

```

```

2     def put(self, id):
3         try:
4             body = {
5                 "id_int": int(id),
6                 "type": request.form.get('type', None),
7                 "name": request.form.get('name', None),
8                 "description": request.form.get('description', None),
9                 "producer": request.form.get('producer', None),
10                "protein": request.form.get('protein', None),
11                "carbohydrate": request.form.get('carbohydrate', None),
12                "min_expired_period": request.form.get(
13                    'min_expired_period', None),
14                "max_expired_period": request.form.get(
15                    'max_expired_period', None),
16                "image": request.form.get('image', None),
17            }
18            inventory = FeedName.objects.get(id_int = int(id)).update
19            (**body)
20            response = {"message": "success update feed inventory", "data": body}
21            response = json.dumps(response, default=str)
22            return Response(response, mimetype="application/json",
23                            status=200)
24        except Exception as e:
25            response = {"message": str(e)}
26            response = json.dumps(response, default=str)
27            return Response(response, mimetype="application/json",
28                            status=400)

```

- Mengambil semua data merk pakan (HTTP Method - DELETE)

```

1     class FeedNameApi(Resource):
2         def delete(self, id):
3             try:
4                 inventory = FeedName.objects.get(id_int = int(id)).delete()
5                 response = {"message": "success delete feed name on
6                 inventory"}
7                 response = json.dumps(response, default=str)
8                 return Response(response, mimetype="application/json",
9                             status=200)
10            except Exception as e:
11                response = {"message": str(e)}
12                response = json.dumps(response, default=str)

```

```

11         return Response(response, mimetype="application/json",
12                         status=400)

```

4. Controller HTTP pada Flutter

- Mengambil semua data merk pakan (HTTP Method - GET)

```

1     Future getPakanNameData(String type, Function() doAfter) async {
2
3         feedNameList.value.data!.clear();
4
5         listPakanName.clear();
6
7         isLoadingName.value = true;
8
9
10        SharedPreferences prefs = await SharedPreferences.getInstance()
11
12        ;
13
14        String token = prefs.getString('token').toString();
15        var headers = {'Authorization': 'Bearer $token'};
16
17
18        final response = await http
19
20            .get(Uri.parse('${Urls.feedNameList}?type=$type'), headers:
21
22            headers);
23
24
25        try {
26
27            if (response.statusCode == 200) {
28
29                InventarisPakanNameModel res =
30
31                InventarisPakanNameModel.fromJson(jsonDecode(response.body)
32
33            );
34
35
36            feedNameList.value = res;
37
38
39            if (feedNameList.value.data!.isNotEmpty) {
40
41                for (var i in feedNameList.value.data!) {
42
43                    listPakanName.add({
44
45                        'id': i.idInt,
46
47                        'feed_name_id': i.sId,
48
49                        'feed_name': i.name,
50
51                    });
52
53                }
54
55
56                selectedPakan.value = listPakanName[0];
57
58            }
59
60        }
61
62        doAfter();

```

```

33     } catch (e) {
34         throw Exception(e);
35     }
36     isLoadingName.value = false;
37 }
38

```

- Mengambil spesifik data merk pakan (HTTP Method - GET)

```

1     Future getDetailPakanNameData(int id, Function() doAfter) async {
2         isLoadingNameDetail.value = true;
3
4         final response = await http.get(Uri.parse('${Urls.feedNameList
5             }/$id'));
6
7         try {
8             if (response.statusCode == 200) {
9                 DetailInventarisPakanNameModel res =
10                    DetailInventarisPakanNameModel.fromJson(jsonDecode(response
11             .body));
12
13             feedCategory.value = res.data!.type.toString();
14             name.text = res.data!.name.toString();
15             producer.text = res.data!.producer.toString();
16             desc.text = res.data!.description.toString();
17             protein.text = res.data!.protein.toString();
18             carbo.text = res.data!.carbohydrate.toString();
19             minExp.text = res.data!.minExpiredPeriod.toString();
20             maxExp.text = res.data!.maxExpiredPeriod.toString();
21         }
22         doAfter();
23     } catch (e) {
24         throw Exception(e);
25     }
26     isLoadingNameDetail.value = false;
}

```

- Menambahkan semua data merk pakan (HTTP Method - POST)

```

1     Future postPakanNameData(Function() doAfter) async {
2         var map = <String, dynamic>{};
3
4         SharedPreferences prefs = await SharedPreferences.getInstance()
;

```

```

5     String token = prefs.getString('token').toString();
6     var headers = {'Authorization': 'Bearer $token'};
7
8     map['type'] = feedCategory.value;
9     map['name'] = name.text;
10    map['description'] = desc.text == '' ? '-' : desc.text;
11    map['producer'] = producer.text;
12    map['protein'] = protein.text == '' ? '0' : protein.text;
13    map['carbohydrate'] = feedCategory.value == 'Industri'
14        ? '50'
15        : carbo.text == ''
16            ? '0'
17            : carbo.text;
18    map['min_expired_period'] = minExp.text == '' ? '0' : minExp.
19     text;
20    map['max_expired_period'] = maxExp.text == ''
21        ? (int.parse(minExp.text == '' ? '0' : minExp.text) * 2).
22     toString()
23        : maxExp.text;
24    map['image'] = image.value;
25
26    isLoadingPost.value = true;
27
28    try {
29      inspect(map);
30      await http.post(
31        Uri.parse(Urls.feedNameList),
32        body: map,
33        headers: headers,
34      );
35      doAfter();
36    } catch (e) {
37      inspect(e);
38      throw Exception(e);
39    }
40    isLoadingPost.value = false;
}

```

- Memperbarui spesifik data merk pakan (HTTP Method - PUT)

```

1 Future updatePakanNameData(int id, Function() doAfter) async {
2   var map = <String, dynamic>{};
3

```

```

4     map['type'] = feedCategory.value;
5     map['name'] = name.text;
6     map['description'] = desc.text == '' ? '-' : desc.text;
7     map['producer'] = producer.text;
8     map['protein'] = protein.text == '' ? '0' : protein.text;
9     map['carbohydrate'] = feedCategory.value == 'Industri'
10    ? '50'
11    : carbo.text == ''
12    ? '0'
13    : carbo.text;
14     map['min_expired_period'] = minExp.text == '' ? '0' : minExp.
15     text;
16     map['max_expired_period'] = maxExp.text == ''
17    ? (int.parse(minExp.text == '' ? '0' : minExp.text) * 2).
18     toString()
19    : maxExp.text;
20     map['image'] = image.value;
21
22     isLoadingPost.value = true;
23
24     try {
25       inspect(map);
26       await http.put(
27         Uri.parse('${Urls.feedNameList}/$id'),
28         body: map,
29       );
30       doAfter();
31     } catch (e) {
32       inspect(e);
33       throw Exception(e);
34     }
35     isLoadingPost.value = false;
}

```

- Menghapus spesifik data merk pakan (HTTP Method - DELETE)

```

1   Future deletePakanName(int id, Function() doAfter) async {
2     isLoadingDelete.value = true;
3     try {
4       await http.delete(
5         Uri.parse(
6           '${Urls.feedNameList}/$id',
7         ),

```

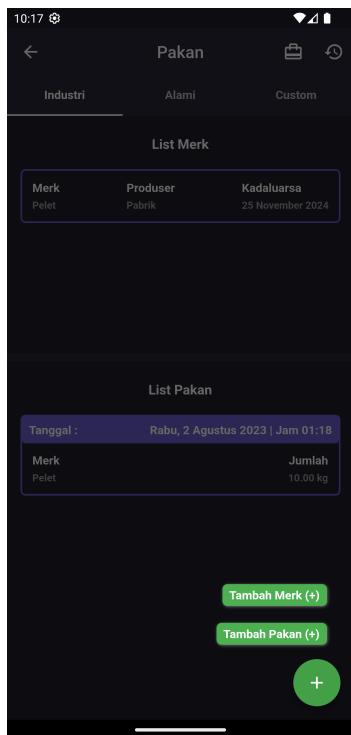
```

8     headers: {
9       'Content-Type': 'application/json; charset=UTF-8',
10      },
11    );
12    doAfter();
13  } catch (e) {
14    throw Exception(e);
15  }
16  isLoadingDelete.value = false;
17}
18

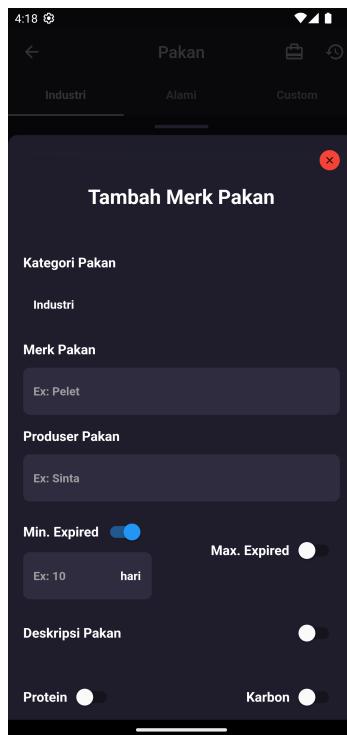
```

5. Tampilan pada Flutter

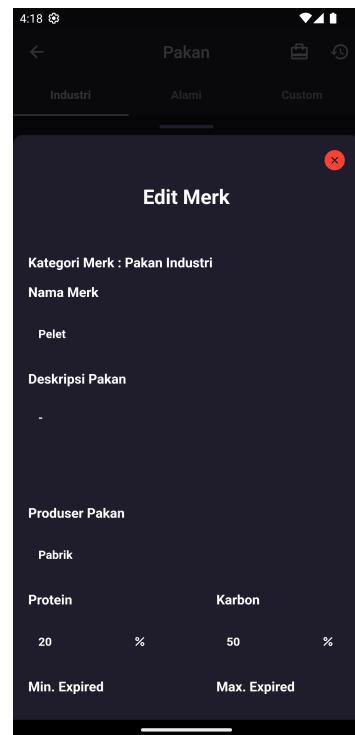
- Halaman Input Merk Pakan & Detail Merk Pakan



Gambar 4.57: Halaman Menu Inventaris Pakan



Gambar 4.58: Halaman Merk Pakan



Gambar 4.59: Halaman Detail Merk Pakan

Pada halaman inventaris pakan, terdapat tombol di pojok kanan bawah yang berfungsi untuk menampilkan opsi antara input pakan dan input merk.

Dalam skenario penambahan pakan, model merk pakan harus diisi terlebih dahulu karena pada saat input data inventaris pakan diperlukan data dari merk pakan tersebut.

6. Integrasi halaman Input Treatment dengan data inventaris suplemen

Pada model backend, model dari PondTreatment berubah menjadi seperti berikut.

```

1      class PondTreatment(db.Document):
2          treatment_type_option = ("ringan", "berat", "pergantian air")
3          farm_id = db.ReferenceField(Farm, required=True)
4
5          probiotic_culture_id = db.ReferenceField(SuplemenInventory, default=
6              None)
7          carbon_id = db.ReferenceField(SuplemenInventory, default=None)
8          salt_id = db.ReferenceField(SuplemenInventory, default=None)
9
10         pond_id = db.ReferenceField(Pond, required=True)
11         pond_activation_id = db.ReferenceField(PondActivation, required=True)
12         treatment_type = db.StringField(
13             required=True, choices=treatment_type_option)
14         water_change = db.IntField()
15         salt = db.FloatField(default=0.0)
16         probiotic_culture_name = db.StringField(default='')
17         probiotic_culture = db.FloatField(default=0.0)
18         carbohydrate = db.FloatField(default=0.0)
19         carbohydrate_type = db.StringField(default="")
20         description = db.StringField(default="")
21         treatment_at = db.DateTimeField(default=datetime.datetime.now)
22         created_at = db.DateTimeField(default=datetime.datetime.now)
23         updated_at = db.DateTimeField(default=datetime.datetime.now)
```

Pada model tersebut, variabel probiotic_culture_id, carbon_id, salt_id didapatkan dari tabel inventaris suplemen. Dalam hal ini, model tabel PondTreatment ini nantinya akan mencatat jenis dan jumlah suplemen yang digunakan.

Perubahan juga terjadi pada bagian Method POST pada class input treatment kolam. Method tersebut berubah menjadi seperti berikut.

```
1 class PondTreatmentsApi(Resource):
2     @jwt_required()
3     def post(self):
4         try:
5             current_user = get_jwt_identity()
6             farm = str(current_user['farm_id'])
7
8             pond_id = request.form.get("pond_id", None)
9             pond = Pond.objects.get(id=pond_id)
10            if pond['isActive'] == False:
11                response = {"message": "pond is not active"}
12                response = json.dumps(response, default=str)
13                return Response(response, mimetype="application/json", status
14=400)
15
16            pond_activation = PondActivation.objects(
17                pond_id=pond_id, isFinish=False).order_by('-activated_at').first()
18
19            treatment_type = request.form.get("treatment_type", None)
20            if treatment_type == "berat":
21
22                fishes = request.form.get("fish", "[]")
23                fishes = json.loads(fishes)
24
25                body = {
26
27                    "pond_id": pond_id,
28                    "farm_id": farm,
29                    "pond_activation_id": pond_activation.id,
30                    "treatment_type": treatment_type,
31                    "water_change": 100,
32                    "description": request.form.get("description", None),
33                }
34
35                pondtreatment = PondTreatment(**body).save()
36
37                id = pondtreatment.id
38
39                # update activation and pond
40
41                pond_deactivation_data = {
42
43                    "isFinish": True,
44                    "total_fish_harvested": request.form.get("total_fish_harvested
45", None),
46
47                    "total_weight_harvested": request.form.get("total_weight_harvested",
48 None),
49
50                    "deactivated_at": request.form.get("deactivated_at", datetime.
51 datetime.now()),
52
53                    "deactivated_description": "karantina total"
54                }
55
56                pond_activation = PondActivation.objects(
57
58                    pond_id=id,
59                    isFinish=True,
60                    total_fish_harvested=id,
61                    total_weight_harvested=id,
62                    deactivated_at=datetime.now(),
63                    deactivated_description="karantina total"
64                )
65
66                pond_activation.save()
67
68            else:
69                response = {"message": "treatment type not found"}
70                response = json.dumps(response, default=str)
71                return Response(response, mimetype="application/json", status
72=400)
73
74        except Exception as e:
75            print(e)
76            response = {"message": "error occurred while processing the request"}
77            response = json.dumps(response, default=str)
78            return Response(response, mimetype="application/json", status
79=500)
```

```
39         pond_id=pond_id, isFinish=False).order_by('-activated_at')
40     first()
41         pond_activation.update(**pond_deactivation_data)
42         pond.update(**{"isActive": False, "status": "Panen"})
43         for fish in fishes:
44             # save fish log
45             data = {
46                 "pond_id": pond_id,
47                 "pond_activation_id": pond_activation.id,
48                 "type_log": "deactivation",
49                 "fish_type": fish['type'],
50                 "fish_amount": fish['amount'],
51                 "fish_total_weight": fish['weight']
52             }
53             # total_fish_harvested += fish['amount']
54             # total_weight_harvested += fish['weight']
55             fishlog = FishLog(**data).save()
56             print(data)
57
58     elif treatment_type == "ringan":
59
60         prob_id = request.form.get("probiotic_culture_id", None)
61         carb_id = request.form.get("carbon_id", None)
62         salt_id = request.form.get("salt_id", None)
63
64         body = {
65             "pond_id": pond_id,
66             "farm_id": farm,
67             "probiotic_culture_id": prob_id ,
68             "pond_activation_id": pond_activation.id,
69             "treatment_type": treatment_type,
70             "description": request.form.get("description", None),
71             "water_change": request.form.get("water_change", 0),
72             "salt": request.form.get("salt", None),
73             "probiotic_culture_name": request.form.get("probiotic_culture_name", None),
74             "probiotic_culture": request.form.get("probiotic_culture", None),
75             "carbohydrate": request.form.get("carbohydrate", None),
76             "carbohydrate_type": request.form.get("carbohydrate_type", None),
77             "treatment_at": request.form.get("treatment_at", datetime.datetime.now())
78         }
79
80     return render_template('pond.html', ponds=ponds, farms=farms, treatments=treatments)
```

```

78         get_suplemen_by_prob = SuplemenInventory.objects.get(id=prob_id)
79         get_suplemen_by_prob.amount -= float(request.form.get(
80             "probiotic_culture", None))
81         get_suplemen_by_prob.save()
82
83         if carb_id != None:
84             body['carbon_id'] = carb_id
85             get_suplemen_by_carb = SuplemenInventory.objects.get(id=
86             carb_id)
87             get_suplemen_by_carb.amount -= float(request.form.get(
88                 "carbohydrate", None))
89             get_suplemen_by_carb.save()
90
91         if salt_id != None:
92             body['salt_id'] = salt_id
93             get_suplemen_by_salt = SuplemenInventory.objects.get(id=
94             salt_id)
95             get_suplemen_by_salt.amount -= float(request.form.get("salt",
96             None))
97             get_suplemen_by_salt.save()
98
99         pondtreatment = PondTreatment(**body).save()
100        id = pondtreatment.id
101        elif treatment_type == "pergantian air":
102            body = {
103                "pond_id": pond_id,
104                "pond_activation_id": pond_activation.id,
105                "treatment_type": treatment_type,
106                "water_change": request.form.get("water_change", 0)
107            }
108            pondtreatment = PondTreatment(**body).save()
109            id = pondtreatment.id
110        else:
111            response = {
112                "message": "treatment type just allow ['ringan','berat']"}
113            response = json.dumps(response, default=str)
114            return Response(response, mimetype="application/json", status
115            =400)
116            response = {
117                "message": "success add data pond treatment", "id": id}
118            response = json.dumps(response, default=str)
119            return Response(response, mimetype="application/json", status=200)
120        except Exception as e:
121            response = {"message": str(e)}

```

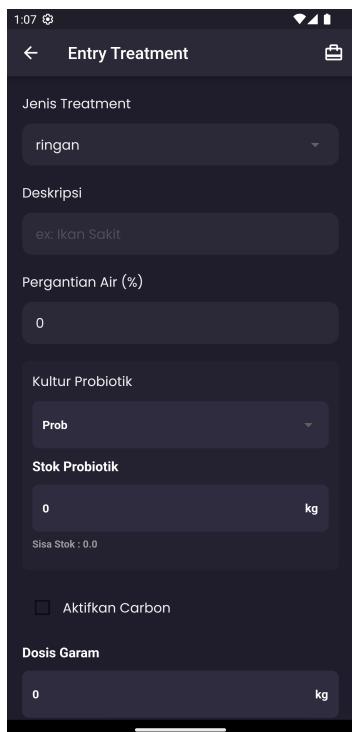
```

116     response = json.dumps(response, default=str)
117     return Response(response, mimetype="application/json", status=400)
118

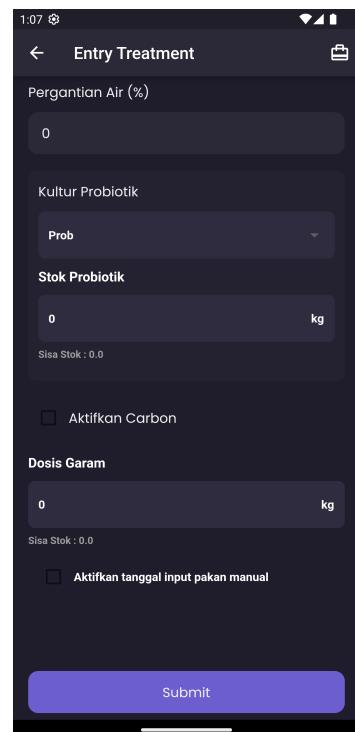
```

Pada method tersebut, terdapat pengambilan data dari inventaris suplemen agar jumlah stok dari inventaris suplemen dapat terupdate jika fitur treatment kolam dijalankan.

Berikut merupakan tampilan dari fitur treatment kolam yang telah diperbarui.



Gambar 4.60: Halaman Treatment Kolam



Gambar 4.61: Halaman Treatment Kolam

Pada halaman tersebut, dapat dilihat bagian pada kultur probiotik, karbon, serta dosis garam yang stoknya diambil dari inventaris suplemen.

Berikut merupakan fungsi entry treatment pada Flutter setelah diubah.

```

1 Future<void> postFishGrading(BuildContext context, Function doInPost)
2   async {
3     bool value = await TreatmentService().postPondTreatment(
4       pondId: pond.id,

```

```

4     prob_id: supState.probID.value,
5     carb_id: supState.carbID.value,
6     salt_id: supState.saltID.value,
7     type: typeController.selected.value,
8     probiotic_name: supState.selectedCultureProbiotik.value['suplemen_name
'],
9     probiotic: probioticController.value.text,
10    desc: descController.value.text,
11    water: waterController.value.text,
12    carbohydrate:
13      supState.carbCheck.value ? carbonController.value.text : '0',
14    carbohydrate_type: supState.carbCheck.value
15      ? supState.selectedCarbon.value['suplemen_name']
16      : 'Tidak ada',
17    salt: saltController.value.text,
18  );
19
20  await supState.postHistorySuplemenData(
21    supState.pondName.value,
22    buildJsonTreatment(),
23    supState.selectedUsedDate.value,
24    () => null,
25  );
26  doInPost();
27 }
28

```

Fungsi diatas akan berjalan jika user menekan tombol Submit pada halaman entry treatment. Pada fungsi entry treatment tersebut, terdapat dua jenis method POST. Method pertama digunakan untuk mengirimkan data entry treatment kepada backend dan method kedua digunakan untuk mengirimkan detail penggunaan treament yang nantinya akan masuk kedalam tabel riwayat pemakaian suplemen.

7. Integrasi harga jual minimum ikan pada fitur Panen

Setelah semua sistem inventaris selesai, dapat dibuat fungsi untuk menentukan harga jual minimum ikan. Berikut merupakan code dari penentuan harga minimum jual ikan.

```

1   Future getAllAssetData(
2       String first,
3       String last,
4       Function() doAfter,
5   ) async {
6       assetList.value.data!.clear();
7       var assetPrice = 0;
8
9       SharedPreferences prefs = await SharedPreferences.getInstance();
10      String token = prefs.getString('token').toString();
11      var headers = {'Authorization': 'Bearer $token'};
12
13      final response = await http.get(
14          Uri.parse('${Urls.invAsset}?start_date=$first&end_date=$last'),
15          Uri.parse(Urls.invAsset),
16          headers: headers,
17      );
18
19      try {
20          if (response.statusCode == 200) {
21              InventarisAssetModel res =
22                  InventarisAssetModel.fromJson(jsonDecode(response.body));
23
24              assetList.value = res;
25
26              if (assetList.value.data!.isNotEmpty) {
27                  for (var i in assetList.value.data!) {
28                      assetPrice += i.price!;
29                  }
30              }
31
32              doAfter();
33          }
34      } catch (e) {
35          throw Exception(e);
36      }
37
38      return (assetPrice / (60 * pondController.ponds.length)).round();
39  }
40
41  Future getAllElectricData(
42      String first, String last, Function() doAfter) async {
43      var filteredPond = [];
44      var electricPrice = 0;

```

```
45
46     electricList.value.data!.clear();
47     filteredPond.clear();
48
49     SharedPreferences prefs = await SharedPreferences.getInstance();
50     String token = prefs.getString('token').toString();
51     var headers = {'Authorization': 'Bearer $token'};
52
53     final response = await http.get(
54         Uri.parse('${Urls.invElect}?start_date=$first&end_date=$last'),
55         headers: headers);
56
57     try {
58         if (response.statusCode == 200) {
59             InventarisListrikModel res =
60                 InventarisListrikModel.fromJson(jsonDecode(response.body));
61
62             electricList.value = res;
63
64             if (electricList.value.data!.isNotEmpty) {
65                 for (var i in electricList.value.data!) {
66                     electricPrice += i.price!;
67                 }
68             }
69             doAfter();
70         }
71     } catch (e) {
72         throw Exception(e);
73     }
74
75     for (var i in pondController.ponds) {
76         if (i.isActive!) {
77             filteredPond.add(i);
78         }
79     }
80
81     return (electricPrice / filteredPond.length).round();
82 }
83
84 Future getHistorySuplemenData(
85     String firstDate,
86     String lastDate,
87     String pondName,
88     Function() doAfter,
```

```

89     ) async {
90         suplemenHistoryList.value.data!.clear();
91         var suplemenPrice = 0;
92
93         SharedPreferences prefs = await SharedPreferences.getInstance();
94         String token = prefs.getString('token').toString();
95         var headers = {'Authorization': 'Bearer $token'};
96
97         final response = await http.get(
98             Uri.parse(
99                 '${Urls.suplemenSch}?start_date=$firstDate&end_date=$lastDate&
100                pond_name=$pondName'),
101                headers: headers,
102                );
103
104        try {
105            if (response.statusCode == 200) {
106                HistorySuplemenModel res =
107                    HistorySuplemenModel.fromJson(jsonDecode(response.body));
108
109                suplemenHistoryList.value = res;
110
111                if (suplemenHistoryList.value.data!.isNotEmpty) {
112                    for (var i in suplemenHistoryList.value.data!) {
113                        suplemenPrice +=
114                            ((i.usage! / i.originalAmount!) * i.suplemen!.price!).round();
115                    }
116
117                    doAfter();
118                }
119            } catch (e) {
120                throw Exception(e);
121            }
122
123            return suplemenPrice;
124        }
125
126        Future getHistoryFeedData(String firstDate, String lastDate, String
127            pondName,
128            Function() doAfter) async {
129            feedHistoryList.value.data!.clear();
130            var feedPrice = 0;

```

```

131     SharedPreferences prefs = await SharedPreferences.getInstance();
132     String token = prefs.getString('token').toString();
133     var headers = {'Authorization': 'Bearer $token'};
134
135     final response = await http.get(
136         Uri.parse(
137             '${Urls.feedSch}?start_date=$firstDate&end_date=$lastDate&
138             pond_name=$pondName'),
139             headers: headers);
140
141     try {
142         if (response.statusCode == 200) {
143             HistoryFeedModel res =
144                 HistoryFeedModel.fromJson(jsonDecode(response.body));
145
146             feedHistoryList.value = res;
147
148             if (feedHistoryList.value.data!.isNotEmpty) {
149                 for (var i in feedHistoryList.value.data!) {
150                     feedPrice +=
151                         ((i.usage! / i.originalAmount!) * i.feed!.price!).round();
152                 }
153             }
154
155             doAfter();
156         }
157     } catch (e) {
158         throw Exception(e);
159     }
160     return feedPrice;
161 }
162
163 Future getHistorySeedData(String firstDate, String lastDate, String
164 pondName,
165     Function() doAfter) async {
166     seedHistoryList.value.data!.clear();
167
168     SharedPreferences prefs = await SharedPreferences.getInstance();
169     String token = prefs.getString('token').toString();
170     var headers = {'Authorization': 'Bearer $token'};
171
172     final response = await http.get(
173         Uri.parse(
174             '${Urls.seedSch}?start_date=$firstDate&end_date=$lastDate&pond_name=

```

```

    $pondName') ,
173         headers: headers,
174     );
175
176     try {
177         if (response.statusCode == 200) {
178             HistorySeedModel res =
179                 HistorySeedModel.fromJson(jsonDecode(response.body));
180
181             seedHistoryList.value = res;
182
183             if (seedHistoryList.value.data!.isNotEmpty) {
184                 for (var i in activation.fishLive!) {
185                     for (var j in seedHistoryList.value.data!) {
186                         if (i.fishId == j.fishSeedId) {
187                             if (i.type == "patin") {
188                                 patinPrice.value += ((j.usage! / j.originalAmount!) *
189                                     j.seed!.price! *
190                                     j.originalAmount!)
191                                     .round();
192                             }
193                             if (i.type == "lele") {
194                                 lelePrice.value += ((j.usage! / j.originalAmount!) *
195                                     j.seed!.price! *
196                                     j.originalAmount!)
197                                     .round();
198                             }
199                             if (i.type == "mas") {
200                                 masPrice.value += ((j.usage! / j.originalAmount!) *
201                                     j.seed!.price! *
202                                     j.originalAmount!)
203                                     .round();
204                             }
205                             if (i.type == "nila hitam") {
206                                 nilaHitamPrice.value += ((j.usage! / j.originalAmount!) *
207                                     j.seed!.price! *
208                                     j.originalAmount!)
209                                     .round();
210                             }
211                             if (i.type == "nila merah") {
212                                 nilaMerahPrice.value += ((j.usage! / j.originalAmount!) *
213                                     j.seed!.price! *
214                                     j.originalAmount!)
215                                     .round();

```

```

216         }
217     }
218     }
219     }
220   }
221
222   doAfter();
223 }
224 } catch (e) {
225   throw Exception(e);
226 }
227 }
228

```

Fungsi-fungsi diatas digunakan untuk mendapatkan data dari masing-masing sistem inventaris. Dari urutan fungsi tersebut, data dari inventaris aset diambil terlebih dahulu kemudian semua pengeluaran dari aset ini akan dibagi 60 sesuai dengan formula penentuan harga minimum ikan yang sudah didiskusikan sebelumnya.

Kemudian, fungsi selanjutnya merupakan pengambilan data inventaris listrik, yang dimana semua jumlah pengeluaran listrik ini akan dibagi berdasarkan jumlah kolam ikan yang masih aktif.

Selanjutnya ada fungsi pengambilan riwayat benih, pakan dan suplemen. Pada fitur riwayat benih, pakan, dan suplemen sudah terekam semua penggunaannya sehingga dari fungsi ini akan didapat jumlah pengeluaran dari masing-masing inventaris pakan, suplemen, dan benih.

Masing-masing fungsi inventaris tersebut difilter berdasarkan tanggal awal mulai musim budidaya dan tanggal dimana musim budidaya akan berakhir (panen).

Setelah semua inventaris sudah didapatkan harganya, kemudian harga ikan tersebut disesuaikan berdasarkan jenis ikan yang ada pada musim budidaya tersebut dikarenakan satu kolam itu bisa menggunakan banyak benih ikan sehingga harga masing-masing benih pastinya berbeda. Fungsi filter ini dapat

dilihat sebagai berikut.

```
1     Future getAllInventory(String firstDate, String lastDate) async {
2         isLoadingInventory.value = true;
3
4         DateTime now = DateTime.now();
5         var currMonth = DateTime.now().month;
6         var currYear = DateTime.now().year;
7         int lastday = DateTime(now.year, now.month + 1, 0).day;
8
9         try {
10
11             var valueA = await getAllAssetData(
12                 firstDate,
13                 lastDate,
14                 () => null,
15             );
16
17             var valueB = await getAllElectricData(
18                 '$currYear-$currMonth-01',
19                 '$currYear-$currMonth-$lastday',
20                 () => null,
21             );
22             var valueC = await getHistorySuplemenData(
23                 firstDate,
24                 lastDate,
25                 pondName.value,
26                 () => null,
27             );
28             var valueD = await getHistoryFeedData(
29                 firstDate,
30                 lastDate,
31                 pondName.value,
32                 () => null,
33             );
34             await getHistorySeedData(
35                 firstDate,
36                 lastDate,
37                 pondName.value,
38                 () => null,
39             );
40
41             for (var i in activation.fishLive!) {
42                 if (i.type == 'lele') {
43                     lelePriceController.text = ConvertToRupiah.formatToRupiah(
```

```

44     ((valueA + valueB + valueC + valueD + lelePrice.value) /
45      activation.fishAmount!)
46      .round(),
47  );
48 }
49 if (i.type == 'mas') {
50 masPriceController.text = ConvertToRupiah.formatToRupiah(
51   ((valueA + valueB + valueC + valueD + masPrice.value) /
52      activation.fishAmount!)
53      .round(),
54  );
55 }
56 if (i.type == 'patin') {
57 patinPriceController.text = ConvertToRupiah.formatToRupiah(
58   ((valueA + valueB + valueC + valueD + patinPrice.value) /
59      activation.fishAmount!)
60      .round(),
61  );
62 }
63 if (i.type == 'nila hitam') {
64 nilaHitamPriceController.text = ConvertToRupiah.formatToRupiah(
65   ((valueA + valueB + valueC + valueD + nilaHitamPrice.value) /
66      activation.fishAmount!)
67      .round(),
68  );
69 }
70 if (i.type == 'nila merah') {
71 nilaMerahPriceController.text = ConvertToRupiah.formatToRupiah(
72   ((valueA + valueB + valueC + valueD + nilaMerahPrice.value) /
73      activation.fishAmount!)
74      .round(),
75  );
76 }
77 }
78 } catch (e) {
79 inspect(e);
80 throw Exception(e);
81 }
82
83 Future.delayed(const Duration(seconds: 2), () {
84 isLoadingInventory.value = false;
85 });
86 }
87

```

Berikut merupakan tampilan dari halaman panen ketika harga ikan sudah terlihat.



Gambar 4.62: Halaman Panen

Setelah harga ikan sudah didapatkan, terdapat fungsi rekap panen yang akan berjalan jika user menekan tombol Panen pada halaman panen.

```

1 Future<void> pondDeactivation(BuildContext context, Function doInPost)
2   async {
3     var fishDataRecap = buildJsonFishRecap();
4
5     double weight = getWeight();
6     if (weight == 0) {
7       showDialog<String>(
      context: context,
    
```

```

8   builder: (BuildContext context) => AlertDialog(
9     title: const Text('Input Error',
10       style: TextStyle(color: Colors.red)),
11     content: const Text(
12       'Input Tidak boleh 0/Kosong',
13       style: TextStyle(color: Colors.white),
14     ),
15     backgroundColor: backgroundColor1,
16     shape: RoundedRectangleBorder(
17       borderRadius: BorderRadius.all(Radius.circular(16.0))),
18     actions: <Widget>[
19       TextButton(
20         onPressed: () => Navigator.pop(context, 'OK'),
21         child: const Text('OK'),
22       ),
23     ],
24   )));
25 } else {
26   isDeactivationProgress.value = true;
27   try {
28     await service.postDeactivation(
29       pondId: pond.id,
30       total_fish_harvested: leleAmount.toInt() +
31         patinAmount.toInt() +
32         masAmount.toInt() +
33         nilahitamAmount.toInt() +
34         nilamerahAmount.toInt(),
35       total_weight_harvested: getWeight().toString(),
36       isFinish: true,
37       fish_harvested: buildJsonFish(),
38       doInPost: doInPost,
39       context: context,
40     );
41
42     await deactivationRecapState.postRecap(
43       pond.id.toString(),
44       fishDataRecap,
45       () => null,
46     );
47     doInPost();
48   } catch (e) {
49     //
50   }
51   isDeactivationProgress.value = false;

```

```

52     }
53 }
54

```

Fungsi tersebut merupakan gabungan dari fungsi deaktivasi kolam atau panen, serta fungsi untuk mengirimkan rekap data panen kedalam fitur pembukuan musim budidaya.

8. Design route dan penerapan Flutter untuk pembukuan musim budidaya beserta harga jual ikan (dalam bentuk RESTful API)

- (a) Design sample route

Berikut merupakan sample route yang sudah dibuat untuk pembukuan musim budidaya.

Group	Endpoint	HTTP Status	Operation	Purpose
Pembukuan	/recap/deactivation?type=	GET	READ	mengambil data pembukuan panen
	/recap/deactivation	POST	CREATE	menginput data pembukuan panen

Gambar 4.63: Sample Route Pembukuan

- (b) Model backend

Berdasarkan sample route tersebut, dapat dibuat model dan class HTTP method pada backend. Berikut model serta class untuk pembukuan musim budidaya :

- Model Pembukuan

```

1      class DeactivationRecap(db.Document):
2          id_int = db.SequenceField(required=True)
3          pond_id = db.ReferenceField(Pond, required=True)
4          farm_id = db.ReferenceField(Farm, required=True)
5          fish_seed_id = db.ReferenceField(SeedInventory, required
6          =True)
7          fish_weight = db.FloatField(required=True)
8          fish_amount = db.IntField(required=True)
9          fish_type = db.StringField(required=True)
10         fish_category = db.StringField(required=True)
11         fish_price = db.IntField(required=True)

```

```

11     created_at = db.DateTimeField(default=datetime.datetime.
now)
12     updated_at = db.DateTimeField(default=datetime.datetime.
now)
13

```

(c) Fungsi-fungsi HTTP Method

- Mengambil semua data pembukuan (HTTP Method - GET)

```

1      class DeactivationRecapApi(Resource):
2          @jwt_required()
3          def get(self):
4              try:
5                  current_user = get_jwt_identity()
6                  farm = str(current_user['farm_id'])
7                  farm_id = ObjectId(farm)
8
9                  start_date = datetime.datetime.strptime(request.args
.get('start_date'), '%Y-%m-%d') if request.args.get('start_date')
else datetime.datetime.strptime("2023-01-01", '%Y-%m-%d')
10                 end_date = datetime.datetime.strptime(request.args.
get('end_date'), '%Y-%m-%d') + datetime.timedelta(days=1) if
request.args.get('end_date') else datetime.datetime.strptime(""
2030-01-01", '%Y-%m-%d')
11
12                 pipeline = [
13                     {
14                         '$match': {
15                             'created_at': {
16                                 '$gte': start_date,
17                                 '$lte': end_date,
18                             },
19                             "farm_id": farm_id,
20                         }
21                     },
22                     {"$sort": {"created_at": 1}},
23                     {'$lookup': {
24                         'from': 'pond',
25                         'let': {"pondid": "$pond_id"},
26                         'pipeline': [
27                             {'$match': {'$expr': {'$eq': ['$id', '$
$pondid']}}}, ],
28                         "$project": {

```

```
29         "_id": 1,
30         "alias": 1,
31         "location": 1,
32         "created_at": 1,
33     } }
34 ],
35     'as': 'pond_detail'
36 },
37 {"$addFields": {
38     "pond_detail": {"$first": "$pond_detail"},
39 },
40 ]
41
42     testing = DeactivationRecap.objects.aggregate(
43 pipeline)
44
45
46     response = json.dumps({
47         'status': 'success',
48         'data': temp,
49     }, default=str)
50
51     return Response(response, mimetype="application/json",
52 ", status=200)
53
54     except Exception as e:
55
56         response = {"message": e}
57
58         response = json.dumps(response, default=str)
59
60         return Response(response, mimetype="application/json",
61 ", status=400)
```

- Membuat data pembukuan (HTTP Method - POST)

```
1      class DeactivationRecapApi(Resource):
2          @jwt_required()
3          def post(self):
4              try:
5                  current_user = get_jwt_identity()
6                  farm = str(current_user['farm_id'])
7
8                  body = {
9                      "pond_id": request.form.get('pond_id'),
10                     "farm_id": farm,
```

```
11     "fish_seed_id": request.form.get('fish_seed_id'),  
12     "fish_weight": request.form.get('fish_weight'),  
13     "fish_amount": request.form.get('fish_amount'),  
14     "fish_type": request.form.get('fish_type'),  
15     "fish_category": request.form.get('fish_category')  
16  
17     }  
18  
19     DeactivationRecap(**body).save()  
20     res = {"message": "success add deactivation recap"}  
21     response = json.dumps(res, default=str)  
22     return Response(response, mimetype="application/json")  
23     ", status=200)  
24  
25     except Exception as e:  
26         response = {"message": e}  
27         response = json.dumps(response, default=str)  
28         return Response(response, mimetype="application/json")  
29     ", status=400)
```

(d) Controller HTTP pada Flutter

- Mengambil semua data pembukuan (HTTP Method - GET)

```
17     if (response.statusCode == 200) {
18
19         DeactivationRecapModel res =
20
21         DeactivationRecapModel.fromJson(jsonDecode(response.
22
23         body));
24
25
26         deactivateRecapList.value = res;
27
28         doAfter();
29     }
30
31     } catch (e) {
32
33         throw Exception(e);
34
35     }
36
37     isLoadingPage.value = false;
38
39 }
```

- Membuat data pembukuan (HTTP Method - POST)

```
1     Future postRecap(String pondId, List fishes, Function()
2       doAfter) async {
3
4         var map = <String, dynamic>{};
5
6
7         SharedPreferences prefs = await SharedPreferences.
8         getInstance();
9
10        String token = prefs.getString('token').toString();
11        var headers = {'Authorization': 'Bearer $token'};
12
13        map['pond_id'] = pondId;
14
15        isLoadingPost.value = true;
16
17        for (var i = 0; i < fishes.length; i++) {
18          map['fish_seed_id'] = fishes[i]['fish_seed_id'];
19          map['fish_type'] = fishes[i]['type'];
20          map['fish_amount'] = fishes[i]['amount'];
21          map['fish_weight'] = fishes[i]['weight'];
22          map['fish_category'] = fishes[i]['fish_category'];
23          map['fish_price'] = fishes[i]['fish_price'];
24
25          inspect(map);
26
27          try {
28            final response = await http.post(
29              Uri.parse(Urls.deactivationRecap),
```

```

25         headers: headers,
26         body: map,
27     );
28     inspect(response);
29     doAfter();
30 } catch (e) {
31     throw Exception(e);
32 }
33 }

34 isLoadingPost.value = false;
35
36 }
37

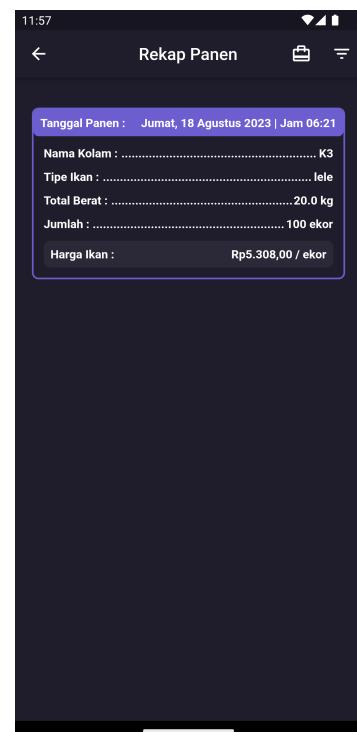
```

(e) Tampilan pada Flutter

- Halaman Pembukuan



Gambar 4.64: Halaman Dashboard



Gambar 4.65: Halaman Pembukuan

Pada halaman dashboard, terdapat tombol buku di pojok kiri atas layar yang akan menavigasikan ke halaman pembukuan. Pada

halaman pembukuan, terdapat list dari riwayat panen dengan informasi detail ikan yang dipanen yang terdiri dari nama kolam, kategori, tipe ikan, total berat, jumlah ikan, dan harga ikan per ekornya.

9. Sprint 5 Review

Hasil review pada Sprint 5 ini adalah review dan testing oleh penulis selaku developer dengan Scrum Master. Setelah dilakukan testing, Scrum Master menyimpulkan bahwa integrasi antara inventaris suplemen dengan treatment kolam sudah berjalan dengan baik. Pada halaman panen juga sudah menampilkan harga minimum ikan yang sudah sesuai berdasarkan kalkulasi formula penentuan harga jual minimum ikan. Pembukuan juga sudah menampilkan rekap data panen dengan informasi ikan dan berjalan dengan baik.

B. Kesimpulan Sprint

Dari semua Sprint yang sudah dilakukan, berikut tabel dari kegiatan pada keseluruhan Sprint.

Tabel 4.6: Sprint 1 Backlog

No	Stories	Task	Status
1	Fitur pencatatan inventaris	<ul style="list-style-type: none"> - Membuat skema database dari pencatatan inventaris - Membuat integrasi skema database dengan skema database sebelumnya - Membuat mockup dari fitur inventaris 	Selesai Selesai Selesai

Tabel 4.7: Sprint 2 Backlog

No	Stories	Task	Status
1	Fitur pencatatan inventaris	- Membuat alur UI/UX dari design aplikasi	Selesai
		- Mengupdate skema database pada inventaris	Selesai

Tabel 4.8: Sprint 3 Backlog

No	Stories	Task	Status
1	Fitur pencatatan inventaris	- Design route untuk inventaris benih (dalam bentuk RESTful API) - Membuat halaman serta Integrasi RESTful API benih dengan Flutter	Selesai
2	Fitur Aktivasi kolam dengan inventaris	- Membuat tabel riwayat pemakaian benih - Design route dan penerapan dengan Flutter untuk riwayat pemakaian benih (dalam bentuk RESTful API)	Selesai

Tabel 4.9: Sprint 4 Backlog

No	Stories	Task	Status
1	Fitur pencatatan inventaris	- Design route dan penerapan pada Flutter untuk inventaris pakan (dalam bentuk RESTful API)	Selesai
		- Design route dan penerapan pada Flutter untuk inventaris suplemen (dalam bentuk RESTful API)	Selesai
		- Design route dan penerapan pada Flutter untuk inventaris listrik (dalam bentuk RESTful API)	Selesai
		- Design route dan penerapan pada Flutter untuk inventaris aset (dalam bentuk RESTful API)	Selesai
		- Design route dan penerapan pada Flutter untuk riwayat pemakaian pakan (dalam bentuk RESTful API)	Selesai
		- Design route dan penerapan pada Flutter untuk riwayat pemakaian suplemen (dalam bentuk RESTful API)	Selesai
2	Fitur Pemberian pakan yang terkoneksi dengan inventaris	- Integrasi data inventaris pakan pada halaman entry pakan	Selesai

Tabel 4.10: Sprint 5 Backlog

No	Stories	Task	Status
1	Fitur pencatatan inventaris	- Design route dan penerapan pada Flutter untuk merk di inventaris pakan (dalam bentuk RESTful API)	Selesai
2	Fitur Treatment kolam yang terkoneksi dengan inventaris	- Integrasi halaman Input Treatment dengan data inventaris suplemen	Selesai
3	Fitur Panen termasuk harga nilai jual ikan	- Integrasi harga jual minimum ikan pada fitur Panen	Selesai
4	Fitur Pembukuan musim budidaya	- Design route dan penerapan Flutter untuk rekapitulasi panen beserta harga jual ikan (dalam bentuk RESTful API)	Selesai

Dari semua Sprint tersebut, dihasilkan aplikasi Aqua Breeding versi kedua dengan penambahan fitur inventaris untuk tracking pengeluaran budidaya serta dapat digunakan untuk menentukan harga dasar penjualan ikan.

Pada tanggal 21 Agustus 2023, dilaksanakan rapat dengan para pembudidaya ikan yang membahas formula penentuan harga dasar berdasarkan inventaris. Dari rapat tersebut, jenis-jenis inventaris yang meliputi inventaris pakan, inventaris suplemen, inventaris listrik, inventaris benih, dan inventaris aset dapat digunakan untuk menentukan harga dasar jual ikan. Namun, terdapat faktor yang tidak dimasukkan yaitu tenaga kerja.

Berdasarkan diskusi dengan stackholder, tenaga kerja tidak dimasukkan kedalam penentuan harga dasar ini sebab harga jual ikan akan melonjak naik. Oleh

karena itu, harga dasar ini hanya berbasis pada pengeluaran yang dikeluarkan oleh pembudidaya.

Berikut merupakan bukti dari diskusi dapat dilihat pada gambar berikut.



Gambar 4.66: Rapat dengan Pembudidaya Ikan

Pada penelitian ini, formula penentuan harga belum di testing kepada para pembudidaya sehingga formula ini masih merupakan hipotesis untuk menyelesaikan masalah pada saat berbudi daya ikan.

C. Pengujian Sistem

Pengujian sistem dilakukan dengan dua tahap yaitu unit testing dan UAT. Pengujian dilakukan ketika task pada Sprint Backlog sudah selesai dilakukan.

1. Unit Testing

Adapun hasil dari unit testing yang telah dilaksanakan kepada tim internal developer dapat dilihat pada tabel-tabel dibawah ini:

Tabel 4.11: Unit testing fitur inventarisasi.

Awal Tabel			
Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Fitur inventaris pakan	✓		Diterima
Fitur inventaris suplemen	✓		Diterima
Fitur inventaris benih	✓		Diterima
Fitur inventaris listrik	✓		Diterima
Fitur inventaris aset	✓		Diterima

Akhir Tabel 4.11

Tabel 4.12: Unit testing integrasi inventarisasi dengan sistem.

Awal Tabel			
Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Aktivasi kolam dengan inventaris	✓		Diterima
Pemberian pakan dengan inventaris	✓		Diterima
Treatment kolam dengan inventaris	✓		Diterima
Panen kolam dengan harga jual minimum ikan	✓		Diterima
Pembukuan panen musim budidaya	✓		Diterima

Akhir Tabel 4.12

2. User Acceptance Test

User Acceptance Test terhadap user dilaksanakan pada tanggal 5 Agustus 2023 secara luring bertempat di Kecamatan Jasinga. Adapun hasil dari UAT yang telah dilaksanakan dapat dilihat pada tabel di bawah ini.

Tabel 4.13: Format *User Acceptance Test*

<i>User Acceptance Test</i>					
No	Acceptance Requirements	Kesesuaian			
		SS	S	TS	STS
1	Fitur inventaris pakan	✓			
2	Fitur inventaris suplemen	✓			
3	Fitur inventaris benih	✓			
4	Fitur inventaris listrik	✓			
5	Fitur inventaris aset	✓			
6	Aktivasi kolam dengan inventaris	✓			
7	Pemberian pakan dengan inventaris	✓			
8	Treatment kolam dengan inventaris		✓		
9	Panen kolam dengan harga jual minimum ikan	✓			
10	Pembukuan panen musim budidaya		✓		

Berdasarkan tabel yang telah di atas, terdapat beberapa fitur yang memerlukan revisi diantaranya:

1. Pada fitur treatment kolam perlu ditambahkan validasi dari masing-masing jenis inventaris suplemen agar dapat melakukan treatment dengan benar.
2. Pada fitur pembukuan panen musim budidaya, masih perlu ditambahkan informasi mengenai data panen selain data harga ikan.

3. Kesimpulan Pengujian

Pengujian aplikasi dilaksanakan dengan dua tahap yaitu unit testing dan UAT. Berdasarkan hasil di atas, seluruh skenario pada unit testing berjalan dengan baik. Namun, berdasarkan hasil UAT terhadap pembudidaya terdapat beberapa masukan seperti fitur treatment kolam dan pembukuan hasil panen.

BAB V

KESIMPULAN DAN SARAN

A. Kesimpulan

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

1. Terbentuknya aplikasi Aqua Breeding versi kedua dengan fitur sistem inventarisasi serta penentuan harga jual minimum ikan. Adapun perancangan aplikasi ini dilakukan dengan metode Scrum dimulai dari tahap penyusunan Product Backlog, Sprint Backlog, dan dikerjakan dalam lima Sprint.
2. Berdasarkan hasil pengujian, seluruh skenario pada unit testing berjalan dengan baik.

B. Saran

Adapun saran untuk penelitian selanjutnya adalah:

1. Berdasarkan diskusi dengan owner, pada versi selanjutnya adalah penambahan fitur transaksi antar pembudidaya ikan yang berguna untuk penjualan ikan agar harga ikan hasil panen dapat digunakan.

DAFTAR PUSTAKA

- Chen, Y. (2020). Implementation of water quality management platform for aquaculture based on big data. *2020 International Conference on Computer Information and Big Data Applications (CIBDA)*.
- Flask (2010). Flask. Diakses pada 30 Mei 2023 dari <https://flask.palletsprojects.com/en/2.3.x/>.
- Flutter (2017). Flutter. Diakses pada 30 Mei 2023 dari <https://flutter.dev/>.
- Hadi, F. P. (2021). Rancang bangun web service dan website sebagai storage engine dan monitoring data sensing untuk budidaya ikan air tawar. *Program Studi Ilmu Komputer Fakultas Matematika Dan Ilmu Pengetahuan Alam Universitas Negeri Jakarta 2021*.
- Hareton K.N. Leung, P. W. W. (1997). A study of user acceptance tests. *Software Quality Journal volume*.
- Ken Schwaber, J. S. (2020). The definitive guide to scrum: The rules of the game. *The Scrum Guide*.
- Lin, Y.-B. (2019). Fishtalk: An iot-based mini aquarium system. *IEEE Access*.
- Maghriza, G. C. (2022). Perancangan frontend aplikasi pendukung teknologi perikanan modern dengan menggunakan framework flutter yang mentarget multi platform. *Program Studi Ilmu Komputer Fakultas Matematika Dan Ilmu Pengetahuan Alam Universitas Negeri Jakarta 2022*.
- MongoDB (2009). Mongodb. Diakses pada 30 Mei 2023 dari <https://www.mongodb.com/>.
- Ouyang, B. (2021). Initial development of the hybrid aerial underwater robotic system (haucs): Internet of things (iot) for aquaculture farms. *IEEE Intenet of Things Journal*.

Rahmanto, A. (2022). Perancangan arsitektur aplikasi budidaya perikanan modern pada backend yang bertanggung jawab dalam melayani transaksi query webservice dengan menggunakan teknologi flask microservice. *Program Studi Ilmu Komputer Fakultas Matematika Dan Ilmu Pengetahuan Alam Universitas Negeri Jakarta 2022.*

Scrum (2010). Scrum. Diakses pada 30 Mei 2023 dari <https://www.scrum.org/>.

Sim, S. (2022). *Dasar-Dasar Manajemen Keuangan (Fundamentals of Financial Management)*. Uwais Inspirasi Indonesia.

Supitriyani (2022). *Management Control*. Media Sains Indonesia.

LAMPIRAN A

LAMPIRAN

A. Transkrip Percakapan

Hari: Rabu

Tanggal: 15 Maret 2023

PL: Penulis

KL: Klien (Pemilik Farm)

PL: Sistem apa yang akan di buat?

KL: Kita akan membuat sistem inventaris pada budidaya perikanan untuk penentuan harga jual minimum ikan

PL: Apa saja requirement dan fitur yang dibutuhkan oleh sistem ini?

KL: Fitur utama yang harus ada adalah sistem inventaris benih, inventaris pakan, inventaris suplemen, inventaris listrik, dan inventaris aset

PL: Bagaimana cara menentukan harga jual minimum ikan dari sistem inventaris tersebut?

KL: Dengan cara menghitung pengeluaran benih, pakan, suplemen, aset serta pengeluaran listrik per kolam aktif selama musim budidaya berjalan. Kemudian, pengeluaran tersebut akan dibagi dengan total ikan ketika masa panen tiba.

PL: Apakah dengan masuknya inventaris aset akan mempengaruhi nilai jual ikan per musim panen?

KL: Dengan masuknya harga aset kedalam perhitungan, harga ikan otomatis akan menjadi tinggi. Untuk itu, harga aset dibagi dengan 5 tahun masa pemakaian karena masa aset dalam inventaris bisa berlangsung lama. Dalam kasus ini diubah menjadi 60 bulan karena musim panen bisa dilakukan tiap bulan.

PL: Dimulai dari manakah pelaksanaan fitur-fitur tersebut?

KL: Dimulai dari inventaris benih

LAMPIRAN B

DAFTAR RIWAYAT HIDUP



Gambar 2.1: Foto Penulis

AKBAR MAULANA ALFATIH, Lahir di Jakarta, 23 Agustus 2002. Anak pertama dari pasangan Bapak Ibnu Akil dan Ibu Sri Ambarwati. Saat ini penulis tinggal di Bambu Hijau Town House, Jl Bambu Hijau Kav. 7, Kel. Cilangkap, Kec. Cipayung, Jakarta Timur.

Riwayat Hidup: Penulis mengawali pendidikan di TK Mutiara Insani pada tahun 2006-2007. Kemudian melanjutkan pendidikan di Madrasah Ibtidaiyah Ruhul Ulum pada tahun 2007-2010, kemudian SDN Cigombong 02 pada tahun 2010-2013. Selanjutnya penulis melanjutkan ke SMPN 222 Jakarta pada tahun 2013-2016. Kemudian melanjutkan pendidikan di SMAN 64 Jakarta pada tahun 2016-2019. Pada tahun 2019 penulis melanjutkan ke Universitas Negeri Jakarta (UNJ) di program studi Ilmu Komputer melalui jalur SNMPTN.