

**APLIKASI AQUABREEDING DENGAN INTEGRASI FITUR
MULTIUSER PEMBUDIDAYA, SISTEM LELANG DAN SISTEM
KEUANGAN SEBAGAI SISTEM INTERKONEKSI PETANI**

Skripsi

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



*Mencerdaskan dan
Memartabatkan Bangsa*

Oleh:
Abdullah Azzam
1313619028

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

2024

LEMBAR PERSETUJUAN HASIL SIDANG SKRIPSI
APLIKASI AQUABREEDING DENGAN INTEGRASI FITUR
MULTIUSER PEMBUDIDAYA, SISTEM LELANG DAN SISTEM
KEUANGAN SEBAGAI SISTEM INTERKONEKSI PETANI

Nama : Abdullah Azzam

No. Registrasi : 1313619028

Nama

Tanda Tangan

Tanggal

Penanggung Jawab

Dekan : Prof. Dr. Muktiningsih N. M.Si.
NIP. 196405111989032001



Wakil Penanggung Jawab

Wakil Dekan I : Dr. Esmar Budi, S.Si., MT.
NIP. 197207281999031002

Ketua : Dr. Ria Arafiah, M.Si.
NIP. 197511212005012004

Sekretaris : Ari Hendarno S.Pd., M.Kom.
NIP. 198811022022031002

Penguji : Drs. Mulyono, M.Kom.
NIP. 196605171994031003

Pembimbing I : Muhammad Eka Suryana, M.Kom.
NIP. 198512232012121002

Pembimbing II : Med Jrzal, M.Kom.
NIP. 197706152003121001

Dinyatakan lulus Ujian Skripsi tanggal: 23 Januari 2024

LEMBAR PERNYATAAN

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul "**Aplikasi Aquabreeding dengan Integrasi Fitur Multiuser Pembudidaya, Sistem Lelang dan Sistem Keuangan sebagai Sistem Interkoneksi Petani**" yang disusun sebagai syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Ilmu Komputer Universitas Negeri Jakarta adalah karya ilmiah saya dengan arahan dari dosen pembimbing.

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

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

Bekasi, 12 Januari 2024



Abdullah Azzam

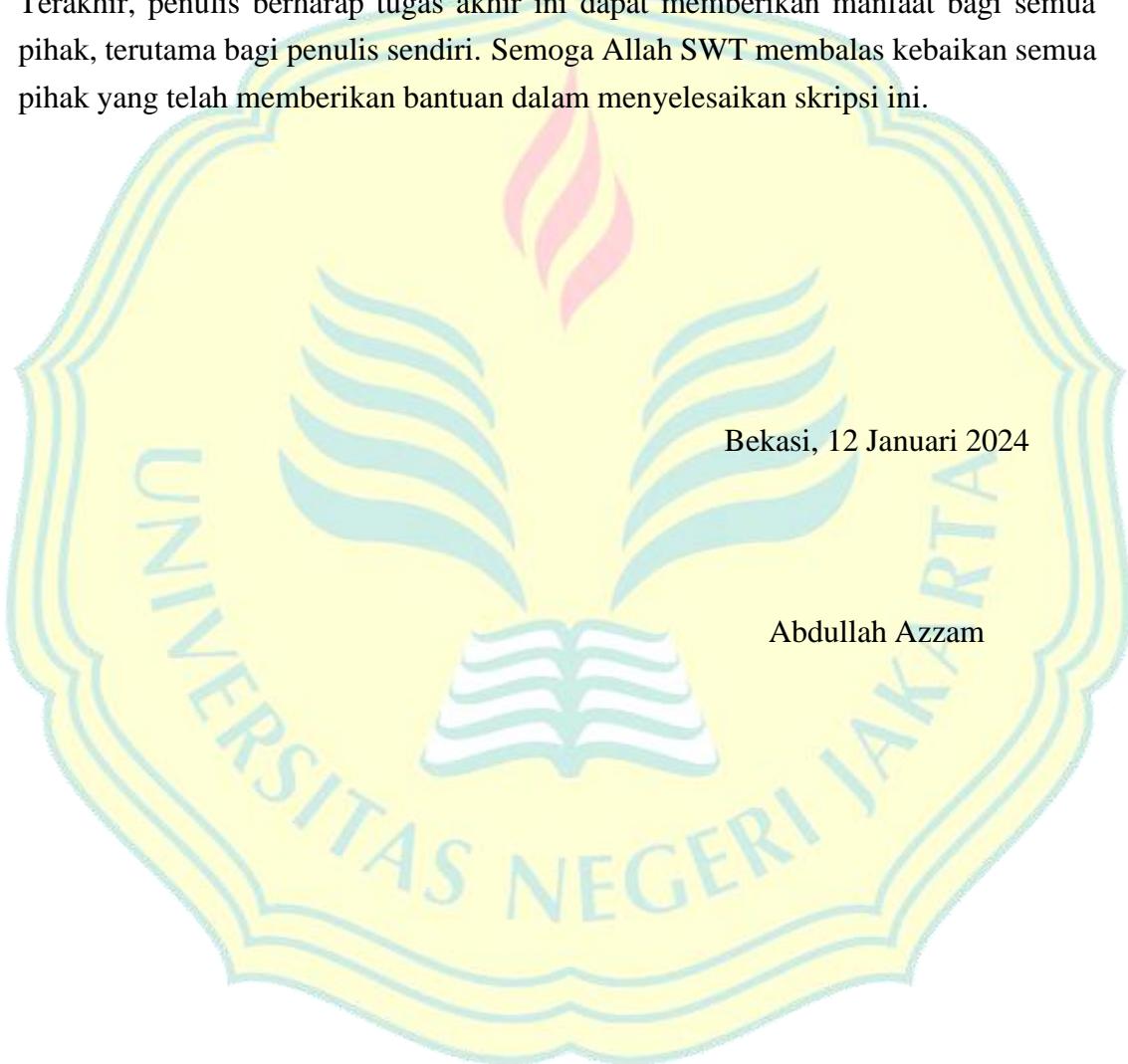
KATA PENGANTAR

Dengan penuh rasa syukur, penulis mengucapkan puji dan syukur kepada Allah SWT, karena atas rahmat dan karunia-Nya, penulis berhasil menyelesaikan skripsi berjudul: *Aplikasi Aquabreeding dengan Integrasi Fitur Multiuser Pembudidaya, Sistem Lelang dan Sistem Keuangan sebagai Sistem Interkoneksi Petani.*

Penulis merasa berterima kasih yang sebesar-besarnya atas keberhasilan dalam menyusun skripsi ini, yang tak dapat dilepaskan dari bantuan berbagai pihak yang dengan tulus dan ikhlas memberikan masukan berarti untuk menyempurnakannya. Dalam kesempatan ini, dengan rendah hati, penulis ingin mengungkapkan rasa terima kasih kepada:

1. Yth. Para petinggi di lingkungan FMIPA Universitas Negeri Jakarta.
2. Yth. Ibu Dr. 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. Kedua orang tua dan adik-adik penulis yang telah mendukung dan memberikan semangat serta doa untuk penulis.
6. Teman-teman Warteg yang selalu mendengarkan dan membantu memberikan saran pada saat proses penulisan skripsi.
7. Teman-teman Program Studi Ilmu Komputer 2019 yang telah memberikan dukungan dan memiliki andil dalam penulisan skripsi ini.
8. Twice dan NewJeans yang telah memberikan dukungan dan menaikkan rasa semangat penulis dalam menuliskan skripsi ini.

Penulis menyadari bahwa penyusunan skripsi ini belum mencapai kesempurnaan karena keterbatasan ilmu dan pengalaman yang dimiliki. Oleh karena itu, penulis dengan senang hati menerima kritik dan saran yang membangun. Terakhir, penulis berharap tugas akhir ini dapat memberikan manfaat bagi semua pihak, terutama bagi penulis sendiri. Semoga Allah SWT membala kebaikan semua pihak yang telah memberikan bantuan dalam menyelesaikan skripsi ini.



*Mencerdaskan dan
Memartabatkan Bangsa*

ABSTRAK

ABDULLAH AZZAM, Aplikasi Aquabreeding dengan Integrasi Fitur Multiuser Pembudidaya, Sistem Lelang dan Sistem Keuangan sebagai Sistem Interkoneksi Petani. Skripsi, Program Studi Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Jakarta. Januari 2024.

Perkembangan teknologi, baik dalam hal hardware maupun software, memberikan dampak positif pada sektor budidaya perikanan air tawar. Sebagai salah satu sektor yang menjanjikan, budidaya ikan menjadi mata pencaharian utama bagi banyak warga Indonesia. Dalam upaya meningkatkan efisiensi dan produktivitas, Aqua Breeding mengusulkan penerapan Internet of Things (IoT) dengan sensor-sensor yang memantau kondisi perairan secara real-time. Meskipun demikian, sektor ini masih dihadapkan pada berbagai tantangan, seperti rendahnya keuntungan pembudidaya, praktik curang dalam transaksi, dan ketimpangan harga penawaran. Untuk mengatasi masalah tersebut, Aqua Breeding versi ketiga dirancang untuk memiliki fitur Multiuser Pembudidaya, Sistem Lelang, dan Sistem Keuangan. Fitur sistem lelang memungkinkan pembudidaya melelang ikan hasil budidaya mereka, dengan pembudidaya lain (pembudidaya penawar) dapat mengajukan penawaran untuk membeli ikan tersebut. Sistem lelang dipilih sebagai metode transaksi jual-beli untuk memberikan pilihan harga yang bervariasi dan kompetitif. Fitur sistem pembayaran memungkinkan pembudidaya penawar membayar jumlah yang ditawarkan jika penawarannya diterima. Selain itu, fitur multiuser disesuaikan dengan perubahan dan penambahan fitur-fitur baru. Metode Scrum digunakan dalam pengembangan aplikasi ini, melibatkan komponen-komponen seperti Product Backlog, Sprint Backlog, Daily Sprint, dan Daily Meet. Hasil akhir dari penelitian ini adalah aplikasi berbasis Android dan REST API.

Kata kunci: panen, lelang, penawaran, transaksi, budidaya ikan.

*Mencerdaskan dan
Memartabatkan Bangsa*

ABSTRACT

ABDULLAH AZZAM, Aqua Breeding Application with Integration of Multiuser Farmer, Auction System, and Financial System as Farmer Interconnection System. Thesis, Computer Science Study Program, Faculty of Mathematics and Natural Sciences, Universitas Negeri Jakarta. January 2024.

The advancement of technology, both in terms of hardware and software, has had a positive impact on the freshwater fish farming sector. As a promising sector, fish farming has become the main livelihood for many Indonesians. In an effort to improve efficiency and productivity, Aqua Breeding proposes the implementation of the Internet of Things (IoT) with sensors that monitor water conditions in real-time. However, this sector still faces various challenges, such as low profits for farmers, fraudulent practices in transactions, and imbalances in bid prices. To address these issues, Aqua Breeding is designed to have Multiuser Farmer, Auction System, and Financial System features as a Farmer Interconnection System. The auction system feature allows farmers to auction their cultivated fish, with other farmers (bidders) able to submit bids to purchase the fish. The auction system is chosen as the method of buying and selling transactions to provide various and competitive pricing options. The payment system feature allows bidder farmers to pay the offered amount if their bid is accepted. Additionally, the multiuser feature is adjusted with the addition of new features. The Scrum method is used in the development of this application, involving components such as Product Backlog, Sprint Backlog, Daily Sprint, and Daily Meet. The final result of this research is an Android-based application and REST API.

Keywords: harvest, auction, bid, transaction, fish farming.

*Mencerdaskan dan
Memartabatkan Bangsa*

DAFTAR ISI

KATA PENGANTAR	v
ABSTRAK	vi
ABSTRACT	vii
DAFTAR ISI	ix
DAFTAR GAMBAR	xi
DAFTAR TABEL	xii
I PENDAHULUAN	1
1.1 Latar Belakang Masalah	1
1.2 Rumusan Masalah	6
1.3 Pembatasan Masalah	6
1.4 Tujuan Penelitian	7
1.5 Manfaat Penelitian	7
II KAJIAN PUSTAKA	8
2.1 Teori Penentuan Harga Pasar Komoditas Pangan	8
2.1.1 Berdasarkan Tiga Proses Penambahan Nilai Ekonomi Komoditas	8
2.1.2 Berdasarkan Struktur Pasar	9
2.2 Teori Penentuan Harga Pangan dari Sisi Produksi dan Bahan Baku	11
2.2.1 Definisi Biaya Produksi	11
2.2.2 Komponen Harga Pokok Produksi	11
2.3 Teori Penentuan Harga Berdasarkan Buying Power Konsumen	13
2.4 Teori Penentuan Harga Berdasarkan Pertemuan <i>Supply-Demand</i>	13
2.5 Teori Penentuan Harga yang Ditentukan Oleh Dominan Power	14
2.6 Konsep Lelang	15
2.7 Karakteristik Lelang	15
2.8 Sistem Keuangan	16
2.9 Teori Graf	16
2.9.1 Teori Graf Algoritma A*	18
2.9.2 Teori Graf tentang Ketetanggaan Lokal (Local Neighborhood)	20
2.10 Maps API	24
2.11 Tingkat Keakuratan Maps API	34
III METODOLOGI PENELITIAN	48
3.1 Keterhubungan Penelitian	48

3.2 Skema Sistem Lelang	49
3.3 Tahapan Penelitian	50
3.4 Analisa Kebutuhan	51
3.5 Perancangan Sistem.....	52
3.6 Pengujian	57
IV HASIL DAN PEMBAHASAN	61
4.1 Perancangan Sistem Dengan Scrum.....	61
4.1.1 Sprint 1.....	61
4.1.2 Sprint 2.....	68
4.1.3 Sprint 3.....	72
4.1.4 Sprint 4.....	80
4.2 Kesimpulan Sprint.....	88
4.3 Pengujian Sistem	94
V KESIMPULAN DAN SARAN	98
5.1 Kesimpulan.....	98
5.2 Saran.....	98
DAFTAR PUSTAKA	99
LAMPIRAN	101
I Object JSON yang dikembalikan oleh <i>endpoint Direction API</i>	101
1.1 OpenStreetMaps	101
1.2 Mapbox.....	108
1.3 HERE Location Services.....	111
II Sprint 2	113
2.1 Fungsi Untuk Menghitung FCR	113
2.2 Query Pipeline untuk Menghitung Survival Rate.....	115
2.3 Fungsi untuk Menyimpan Foto Ketika Panen	122
2.4 Fungsi untuk Menyimpan Data Kualitas Air Harian (pH, DO dan Suhu Air).....	123
2.5 Fungsi untuk Menyimpan Data Kematian Ikan.....	124
2.6 Fungsi agar Aktivasi Kolam dapat Menerima Tipe Data <i>Double</i> pada Ketinggian Air.....	126
III Sprint 3	128
3.1 Fungsi untuk Mengambil Seluruh Data Hasil Panen.....	128
3.2 Fungsi untuk Mengirimkan Data ke Tabel Auction Draft.....	130
3.3 Fungsi untuk Mengambil Seluruh Data dari Tabel Auction Draft	136
3.4 Fungsi untuk Mengirimkan Data ke tabel Auction	138
3.5 Fungsi untuk Mengambil Data dari Tabel Auction	139

3.6 Fungsi untuk Mengirimkan Data ke Tabel Bid	141
3.7 Fungsi untuk Mengambil Data dari Tabel Bid	141
3.8 Fungsi untuk Mengirim Data ke Tabel Transaction.....	143
3.9 Fungsi untuk Mengambil Data dari Tabel Transaction	144
3.10 Fungsi untuk Mengirim Data ke Tabel Request Topup	144
3.11 Fungsi untuk Mengambil Data dari Tabel Request Topup.....	145
3.12 Fungsi untuk Menyetujui Permintaan Topup.....	146
3.13 Fungsi untuk Mengambil Data Jumlah Saldo	147
3.14 Implementasi Route Server	149
IV Transkrip Wawancara Pak Umar	151
4.1 Fitur Rekomendasi Harga.....	151
4.2 Fitur Lelang	152
V Transkrip Wawancara dengan Pak Ending	155
5.1 Fitur Penentuan Harga dan Sistem Lelang	155
5.2 Validasi Fitur Rekomendasi Harga	156
VI Transkrip Wawancara dengan Pak Sobari	157
VII Kesimpulan Wawancara dengan Pak Umar terkait fitur Lelang	159
DAFTAR RIWAYAT HIDUP	160

*Mencerdaskan dan
Memartabatkan Bangsa*

DAFTAR GAMBAR

Gambar 1.1	Presentasi dan Diskusi Mengenai Prospek Penerapan Teknologi di bidang Budidaya Ikan Air Tawar.	4
Gambar 1.2	Presentasi dan Diskusi Mengenai Prospek Penerapan Teknologi di bidang Budidaya Ikan Air Tawar.	5
Gambar 2.1	Tangkapan Layar Hasil Penerapan OpenstreetMap API di Flutter	30
Gambar 2.2	Tangkapan Layar Google Maps untuk Memahami <i>Query Parameter sideOfStreetHint</i>	33
Gambar 2.3	Tangkapan Layar Peta OpenstreetMaps dengan lokasi Stasiun Bekasi	36
Gambar 2.4	Tangkapan Layar Peta OpenstreetMaps yang Menampilkan POI Pom Bensin Total	37
Gambar 2.5	Tangkapan Layar Peta OpenstreetMaps yang Menampilkan POI Kota Cinema Mall Wisma Asri Bekasi	38
Gambar 2.6	Tangkapan Layar Peta OpenstreetMaps yang Menampilkan Pemukiman Warga di Jl. Lele, Kayuringin Jaya, Bekasi	39
Gambar 2.7	Tangkapan Layar Peta Mapbox dengan lokasi Stasiun Bekasi	40
Gambar 2.8	Tangkapan Layar Peta Mapbox yang Menampilkan POI Pom Bensin Total	41
Gambar 2.9	Tangkapan Layar Peta Mapbox yang Menampilkan POI Kota Cinema Mall Wisma Asri Bekasi	42
Gambar 2.10	Tangkapan Layar Peta Mapbox yang Menampilkan Pemukiman Warga di Jl. Lele, Kayuringin Jaya, Bekasi	43
Gambar 2.11	Tangkapan Layar Peta HERE Location Services dengan lokasi Stasiun Bekasi	44
Gambar 2.12	Tangkapan Layar Peta HERE Location Services yang Tidak Menampilkan POI Pom Bensin Total	45
Gambar 2.13	Tangkapan Layar Peta HERE Location Services yang Tidak Menampilkan POI Kota Cinema Mall Wisma Asri Bekasi	46
Gambar 2.14	Tangkapan Layar Peta HERE Location Services yang Menampilkan Pemukiman Warga di Jl. Lele, Kayuringin Jaya, Bekasi	47
Gambar 3.1	Keterhubungan Penelitian	48
Gambar 3.2	Skema Pembudidaya Membuat Lelang	49
Gambar 3.3	Tahapan Penelitian	50
Gambar 3.4	Use Case Aplikasi Aqua Breeding Iterasi Ketiga	52
Gambar 3.5	Skema Database untuk Fitur Pembudidaya Membuat Lelang	55
Gambar 3.6	Integrasi Skema Database Lelang dengan Skema Database Iterasi 2	56

Gambar 4.1	Skema Pembudidaya Membuat Lelang	63
Gambar 4.2	Skema Database untuk Fitur Pembudidaya Membuat Lelang .	65
Gambar 4.3	Integrasi Skema Database Lelang dengan Skema Database Iterasi 2	67
Gambar 4.4	Skema Database Terbaru untuk tabel Auction Draft List, Auction List dan Bid List.	74
Gambar 4.5	Server jft.web.id Pasca Deploy Fitur Lelang dan Multiuser.....	78
Gambar 4.6	Server jft.web.id Pasca Deploy Fitur Sistem Keuangan	79
Gambar 4.7	Pond Page	82
Gambar 4.8	Rekap Panen	82
Gambar 4.9	Menu Page	82
Gambar 4.10	Rekap Panen untuk Membuat Lelang Grosir.....	83
Gambar 4.11	Membuat Lelang Grosir.....	83
Gambar 4.12	Langkah Pertama Membuat Lelang.....	83
Gambar 4.13	Langkah Kedua Membuat Lelang	83
Gambar 4.14	Lelang Saya	83
Gambar 4.15	Detail Lelang	85
Gambar 4.16	Mengajukan Penawaran.....	85
Gambar 4.17	Melihat Penawaran	85
Gambar 4.18	Menerima Penawaran	86
Gambar 4.19	Konfirmasi Menerima Penawaran	86
Gambar 4.20	Riwayat Penawaran	86
Gambar 4.21	Topup.....	87
Gambar 7.1	Pertemuan dengan Pak Umar	159

*Mencerdaskan dan
Memartabatkan Bangsa*

DAFTAR TABEL

Tabel 2.1	Perbedaan Fitur pada masing-masing Maps API	24
Tabel 2.3	Perbedaan Harga pada masing-masing Maps API	25
Tabel 3.1	Product Backlog	53
Tabel 3.2	Sprint Backlog	53
Tabel 3.4	Tabel Route Server yang berisi Route Server untuk Fitur Membuat Lelang	55
Tabel 4.1	Sprint 1 Backlog	61
Tabel 4.3	Tabel Route Server yang Berisi Route untuk Fitur Membuat Lelang	67
Tabel 4.5	Sprint 2 Backlog	69
Tabel 4.7	Sprint 3 Backlog	72
Tabel 4.9	Sprint 4 Backlog	80
Tabel 4.11	Sprint 1 Backlog	88
Tabel 4.13	Sprint 2 Backlog	89
Tabel 4.15	Sprint 3 Backlog	90
Tabel 4.17	Sprint 4 Backlog	92
Tabel 4.19	Unit testing perbaikan fitur aplikasi versi kedua.	94
Tabel 4.20	Unit testing fitur lelang.	95
Tabel 4.21	Unit testing fitur sistem keuangan.	96
Tabel 4.22	Unit testing fitur sistem multiuser.	96

*Mencerdaskan dan
Memartabatkan Bangsa*

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Akhir-akhir ini, perkembangan teknologi baik secara fisik (*hardware*) maupun digital (*software*) melaju secara pesat. Berkat perkembangan teknologi tersebut, banyak prospek-prospek menjanjikan di berbagai sektor kehidupan yang akan sangat terbantu dari segi produktivitas yang juga akan berdampak positif dengan bertambahnya pemasukan. Salah satu sektor yang memiliki prospek yang menjanjikan adalah sektor budidaya perikanan air tawar, sebab budidaya perikanan air tawar cukup mudah untuk dilakukan sehingga banyak warga negara Indonesia yang bermata pencaharian sebagai pembudidaya ikan air tawar. Sehingga, perlu adanya penerapan teknologi agar dapat memudahkan dan meningkatkan efisiensi ketika musim budidaya ikan dimulai. Ketika efisiensi meningkat, maka produktivitas akan meningkat pula, yang akan berdampak pada naiknya pendapatan pembudidaya ikan.

Salah satu penerapan teknologi pada sektor budidaya perikanan yakni penerapan IOT (*Internet of Things*) yang sensornya dapat memantau dan mengirim data-data seperti suhu air, kadar pH pada air, dan kadar DO (*Dissolve Oxygen*) pada air sebab menurut (Pramleonita et al., 2018), data-data tersebut akan menjadi indikator penentu kelayakan tempat hidup ikan secara *real-time* melalui *query webservice* seperti penelitian yang dilakukan oleh (Hadi, 2021) dengan tujuan agar pembudidaya ikan tidak perlu memantau langsung di tambak melainkan bisa di mana saja (tidak terbatas tempat). Hal ini memudahkan pembudidaya karena lebih efisien dari segi waktu dan tenaga.

Namun, penerapan IOT tersebut bukanlah satu-satunya prospek menjanjikan di sektor budidaya perikanan air tawar. Pada tanggal 23 Februari 2023, Bapak Muhammad Eka Suryana dan saudara Gian Chiesa Maghriza selaku perwakilan dari tim riset Aqua Breeding telah melakukan presentasi di hadapan Dinas Peternakan dan Perikanan Kabupaten Bogor mengenai prospek-prospek menjanjikan penerapan teknologi di bidang budidaya perikanan. Presentasi tersebut diawali dengan penyampaian masalah-masalah yang saat ini sedang dihadapi oleh pembudidaya ikan setelah tim riset Aqua Breeding berdiskusi langsung dengan sejumlah

pembudidaya ikan air tawar di Kabupaten Bogor. Permasalahan yang dihadapi oleh pembudidaya ikan air tawar yakni:

1. Pembudidaya mendapat keuntungan yang sedikit pada setiap musim budidaya, bahkan tidak cukup untuk kehidupan sehari-hari. Hal ini disebabkan naiknya harga pakan industri namun tidak diiringi dengan naiknya harga jual ikan.
2. Praktik curang yang terjadi ketika pembudidaya melakukan transaksi dengan tengkulak atau distributor. Dalam praktik ini, tengkulak seringkali mengurangi bobot hasil timbangan, sehingga bobotnya menjadi lebih rendah daripada yang seharusnya. Hal ini dapat berdampak negatif pada jumlah keuntungan yang diterima oleh pembudidaya ikan.
3. Ketimpangan harga penawaran oleh distributor pada petani produsen karena tidak ada data jelas terkait jumlah demand dan besarnya *supply*

Setelah mengetahui tantangan-tantangan yang dihadapi oleh pembudidaya ikan air tawar, Bapak Muhammad Eka Suryana kemudian menyampaikan masalah-masalah tersebut kepada Dinas Peternakan dan Perikanan Kabupaten Bogor dengan harapan agar Dinas tersebut menjadi lebih *aware* dengan banyaknya pembudidaya ikan air tawar yang kesulitan untuk hidup layak kemudian mengambil tindakan yang lebih proaktif dalam menangani isu-isu ini. Setelah menyampaikan permasalahan, Bapak Muhammad Eka Suryana kemudian mengusulkan sebuah solusi (yang merupakan hasil diskusi dengan tim riset Aqua Breeding) untuk mengatasi permasalahan-permasalahan pembudidaya ikan tersebut yakni berupa aplikasi Aqua Breeding yang dapat digunakan dengan mudah melalui (*smartphone*) pembudidaya. Aplikasi Aqua Breeding tersebut akan memiliki tiga fitur utama, yakni:

1. Fitur Manajemen Budidaya Perikanan

Fitur ini akan mencatat aktivitas pembudidaya selama satu musim penuh mulai dari tebar benih ikan hingga panen, seperti mencatat jumlah kolam yang dimiliki pembudidaya, jumlah dan ukuran benih yang ditebar, jumlah pakan yang diberikan, jumlah kematian ikan, jumlah dan bobot panen ikan, dan data-data penting lainnya. Kemudian, dari data-data yang telah terkumpul, akan dihitung konversi pakan/FCR (*Feed Conversion Rate*) yang akan digunakan sebagai patokan untuk menilai efisiensi konsumsi pakan ikan. Semakin kecil nilai FCR berarti ikan dapat bertumbuh mencapai bobot yang

layak untuk dipasarkan dengan penggunaan pakan yang sedikit. Kondisi inilah yang ideal bagi pembudidaya ikan karena jumlah *cost* pembelian pakan bisa diminimalisir sehingga pembudidaya bisa mendapatkan keuntungan yang lebih layak walaupun harga jual ikan tidak naik sekalipun. Fitur ini juga akan mencatat bobot ikan ketika panen dan dapat dijadikan bukti kuat untuk mencegah kecurangan distributor.

2. Fitur Rekomendasi Harga Jual

Fitur ini akan menginventarisasi bahan-bahan dan alat-alat yang digunakan pembudidaya dalam satu musim budidaya seperti jenis dan jumlah benih ikan, jenis dan jumlah suplemen, jenis dan jumlah obat, jumlah pemakaian listrik, dan bahan-bahan lain kemudian menghitung *cost* yang dihabiskan oleh pembudidaya untuk membeli, menyewa, atau membangun bahan-bahan serta alat-alat tersebut. Fitur ini akan membuat pembudidaya mendapatkan harga jual yang lebih layak sehingga pembudidaya tidak akan merugi lagi. Fitur penentuan harga ini juga bisa dijadikan bukti kuat untuk mencegah ketimpangan penawaran harga oleh distributor, dan akan digunakan sebagai harga dasar pada fitur utama ketiga aplikasi Aqua Breeding yakni Sistem Interkoneksi Pembudidaya.

3. Fitur Sistem Interkoneksi Pembudidaya

Fitur ini merupakan titik pertemuan antara besarnya jumlah *supply* dan jumlah *demand* komoditas perikanan air tawar. Pada fitur ini, pembudidaya dapat melelang ikan hasil panen secara daring, dan pembudidaya lain yang menggunakan aplikasi Aqua Breeding akan dapat memberikan penawaran pada ikan yang dilelang tersebut. Transaksi yang terjadi pada fitur ini yakni antar pembudidaya, sebab secara realita ada pembudidaya yang akan merangkap sebagai distributor juga, dan ada kasus nyata di mana pembudidaya membutuhkan ikan dalam jumlah bobot tertentu untuk dipasarkan ke rumah makan namun karena hasil panen miliknya hanya sedikit kemudian pembudidaya tersebut akan membeli ikan dari pembudidaya lain. Ketika seorang pembudidaya akan mengadakan lelang, harga dasar dapat ditentukan dengan mengikuti rekomendasi harga jual. Rekomendasi ini dihitung dengan memperhitungkan total biaya yang dikeluarkan oleh pembudidaya selama satu musim. Kemudian, pembudidaya memiliki kemampuan untuk memodifikasi

harga rekomendasi ini dengan menambahkan margin keuntungan yang diinginkannya. Sebagai contoh, jika harga rekomendasi untuk satu kilogram ikan lele adalah Rp. 20.000, namun pembudidaya ingin mendapatkan keuntungan sebesar Rp. 10.000, maka harga rekomendasi tersebut dapat diubah menjadi Rp. 30.000. Fitur ini juga akan menjadi data nyata dari besar jumlah *supply* dan *demand* komoditas perikanan air tawar.

Setelah pemaparan permasalahan dan solusi oleh Bapak Muhammad Eka Suryana, kemudian Dinas Perikanan dan Peternakan mengakui adanya permasalahan-permasalahan tersebut di lapangan, dan menyetujui rencana untuk mengembangkan aplikasi Aqua Breeding. Dinas tersebut meminta kepada Bapak Muhammad Eka Suryana dan tim riset Aqua Breeding untuk dikirimkan data-data yang dikumpulkan aplikasi agar dapat memantau perkembangan data tersebut, sehingga tim riset memutuskan untuk membuat situs *website* baru untuk memudahkan Dinas memantau data-data dari aplikasi tersebut. Dinas Perikanan dan Peternakan Kabupaten Bogor berjanji akan memasukkan aplikasi Aqua Breeding ke dalam program-program penyuluhan perikanan ketika aplikasi Aqua Breeding telah sepenuhnya selesai dikembangkan.



Gambar 1.1: Presentasi dan Diskusi Mengenai Prospek Penerapan Teknologi di bidang Budidaya Ikan Air Tawar.



Gambar 1.2: Presentasi dan Diskusi Mengenai Prospek Penerapan Teknologi di bidang Budidaya Ikan Air Tawar.

Setelah mendapatkan masukan dan persetujuan dari Dinas Perikanan dan Peternakan Kabupaten Bogor Dengan mempertimbangkan kekurangan-kekurangan yang terdapat pada penelitian-penelitian sebelumnya, tujuan dari penelitian ini adalah untuk menghubungkan pembudidaya secara langsung dengan pengguna/pelanggan dan membuat fitur untuk transaksi antara sesama pembudidaya ikan dan antara pembudidaya dengan distributor terpilih. Dengan demikian, diharapkan pembudidaya akan mendapatkan margin pemasukan yang lebih besar karena rantai pemasaran akan dipangkas, dan mereka dapat menetapkan harga sesuai keinginan. (Junaidi and Maghdahfanti, 2020).

Agar aplikasi bisa menghubungkan pembudidaya dengan pelanggan, maka aplikasi perlu mengetahui lokasi dari setiap pengguna aplikasi. Untuk saling mengetahui lokasi, maka dibutuhkan algoritma yang akan menunjukkan jalan terdekat dari lokasi satu pembudidaya menuju lokasi pembudidaya ikan lain. Algoritma yang akan digunakan pada penelitian kali ini yakni algoritma A*, algoritma ini merupakan algoritma pencarian terbaik yang menggunakan jalur

berbiaya paling rendah yang bisa ditemukan dari node/titik awal ke node/titik tujuan, dengan menjaga prioritas antrian yang diurutkan dari segmen jalur alternatif sepanjang jalan. Algoritma ini menggunakan fungsi “pengetahuan ditambah dengan biaya heuristik” dari sebuah node/titik (biasanya dilambangkan) untuk menentukan urutan di mana pencarian mengunjungi node di pohon. Fungsi “pengetahuan ditambah dengan biaya heuristic” dari sebuah node merupakan penjumlahan dari dua fungsi berikut:

1. Fungsi biaya sebuah jalur masa lalu, yakni jarak yang diukur dari node awal hingga node saat ini.
2. Fungsi biaya sebuah jalur masa depan, yakni jarak “perkiraan heuristic” yang dapat diterima dari node saat ini menuju node tujuan. (Chopde and Nichat, 2013).

Fitur Sistem Lelang akan sangat membantu pembudidaya mendapatkan keuntungan sebesar-besarnya, sebab para penawar akan menawarkan harga setinggi mungkin untuk bisa membeli komoditas ikan hasil panen yang dilelang oleh Pembudidaya. Fitur ini juga akan sangat membantu pembudidaya terhubung dengan pembudidaya ikan lain serta penawar ikan secara digital.

Berdasarkan fitur baru yang telah dijelaskan sebelumnya, aplikasi ini diharapkan dapat membantu para pembudidaya ikan dalam berbudidaya sehingga pembudidaya ikan dapat mendapatkan keuntungan disetiap musim budidayanya.

1.2 Rumusan Masalah

Berdasarkan fokus penelitian yang telah dipaparkan di atas, maka dapat diketahui rumusan masalah penelitian ini adalah "**Bagaimana rancangan Fitur Multiuser Pembudidaya, Sistem Lelang, dan Sistem Keuangan yang akan digunakan untuk transaksi lelang secara daring dalam aplikasi AquaBreeding?**".

1.3 Pembatasan Masalah

Penelitian ini membatasi permasalahan menjadi beberapa bagian seperti berikut agar penelitian ini lebih terfokus dan terarah sesuai dengan tujuan penelitian :

1. Pada aplikasi ini, dibatasi menjadi dua *role*/peran pengguna saja yakni pembudidaya ikan (selanjutnya akan disebut sebagai *breeder*) dan admin/pengelola keuangan.
2. Admin/pengelola keuangan dapat melihat daftar permintaan *topup* dan memilih permintaan *topup* yang hendak disetujui, namun hanya melalui API saja sehingga harus menggunakan Postman.

1.4 Tujuan Penelitian

Penelitian ini dilakukan dengan tujuan untuk membuat aplikasi budidaya ikan modern dengan penerapan fitur sistem lelang.

1.5 Manfaat Penelitian

1. Bagi sektor perikanan

Hasil rancangan aplikasi ini dapat membantu pembudidaya ikan untuk meningkatkan keuntungan melalui skema sistem lelang dimana penawar akan mengajukan penawaran setinggi-tingginya pada ikan yang dilelang.

2. Bagi penulis

Dalam proses perancangan aplikasi, penulis mendapatkan banyak sekali ilmu baru di bidang lelang, transaksi keuangan, multiuser, pengembangan aplikasi Android menggunakan Flutter dan merancang arsitektur REST API menggunakan framework Flask.

3. Bagi Universitas Negeri Jakarta

Penelitian ini dapat dijadikan pedoman untuk penelitian di masa depan serta menjadi panduan bagi mahasiswa Ilmu Komputer tentang cara membuat fitur lelang secara daring, transaksi keuangan mandiri (tanpa bantuan Xendit) dan multiuser pada aplikasi Android.

*Menacerdaskan dan
Memartabatkan Bangsa*

BAB II

KAJIAN PUSTAKA

2.1 Teori Penentuan Harga Pasar Komoditas Pangan

Harga yang ditetapkan pada sebuah barang/komoditas pangan dipengaruhi oleh beberapa faktor. Penetapan harga tidak bisa sembarangan sebab bisa menimbulkan kerugian apabila harga yang ditetapkan terlalu murah, dan akan menyebabkan sepinya pembeli apabila harga yang ditetapkan terlalu mahal. Sehingga, perlu memerhatikan beberapa faktor/landasan agar harga yang ditetapkan sesuai dan bisa menarik pembeli. Berikut beberapa faktor/landasan tersebut.

2.1.1 Berdasarkan Tiga Proses Penambahan Nilai Ekonomi Komoditas

Manusia telah dikenal sebagai makhluk yang selalu mencari keuntungan, terutama dalam aspek finansial. Dalam setiap transaksi perdagangan, termasuk pada berbagai komoditas, umumnya terdapat unsur marjin keuntungan. Dikutip dari (Prastowo et al., 2008), secara konvensional, perilaku pembentukan harga dan marjin keuntungan ini dapat dijelaskan sebagai berikut:

$$P = M + C + \pi \quad (2.1)$$

$$\pi = P - (M + C) \quad (2.2)$$

dimana :

1. P merupakan harga jual
2. M merupakan biaya input atau modal awal yang meliputi pembelian bahan baku untuk produksi, termasuk barang mentah (belum diolah dan berbentuk dasar) serta barang setengah jadi (telah diolah tetapi belum menjadi bentuk akhir barang).
3. C adalah biaya penambahan nilai, meliputi:
 - (a) Biaya pengolahan barang, biasanya merubah bentuk baik dari mentah menjadi setengah jadi, maupun bentuk setengah jadi menjadi barang jadi.

- (b) Biaya penyimpanan digunakan untuk meningkatkan nilai barang dari perspektif perbedaan waktu, seperti kasus anggur yang nilai jualnya akan lebih tinggi apabila disimpan dalam jangka waktu yang lebih lama dibandingkan dengan anggur yang disimpan hanya dalam waktu singkat.
- (c) Biaya distribusi barang untuk menambah nilai

4. π merupakan margin keuntungan

Manusia akan memperoleh keuntungan melalui tiga bentuk proses penambahan nilai ekonomi pada suatu barang. Proses perubahan bentuk dan penyimpanan barang dapat diatur urutannya sesuai kebutuhan, tetapi proses distribusi tetap menjadi tahap puncak karena itu yang menghubungkan barang jadi dengan konsumen. Proses distribusi juga merupakan satu-satunya proses yang pasti untuk meningkatkan nilai ekonomi suatu barang. Tidak semua barang atau komoditas melalui proses perubahan bentuk dan penyimpanan, seperti beberapa barang pertanian dan peternakan, yang dibeli oleh konsumen secara mentah untuk diolah menjadi makanan. Barang-barang tersebut akan membekukan bila disimpan terlalu lama karena sifat perishable-nya. Dengan demikian, efisiensi proses distribusi barang atau komoditas sangat mempengaruhi harga yang dibayar oleh konsumen.

Efisiensi distribusi komoditas dapat ditentukan oleh dua faktor. Ada dua faktor utama yang mempengaruhi efisiensi kegiatan distribusi menuju konsumen. Pertama, panjang rantai distribusi yang harus dilalui dan besarnya harga yang ditetapkan oleh setiap mata rantai distribusi, karena setiap mata rantai akan mengambil keuntungan. Semakin pendek rantai distribusi dan semakin rendah harga yang ditetapkan di setiap mata rantai, maka semakin efisien kegiatan distribusinya. Faktor kedua adalah kondisi sektor transportasi. Gangguan dalam sektor transportasi, seperti terbatasnya moda transportasi yang tersedia, kerusakan jalan, atau gangguan akibat bencana alam, akan meningkatkan biaya dan durasi pengiriman barang. Biaya yang meningkat akibat gangguan-gangguan tersebut akan mempengaruhi harga di tingkat konsumen. Hal yang sama berlaku untuk durasi pengiriman barang, semakin lama barang sampai di tingkat konsumen, nilai barang tersebut akan menurun, terutama jika merupakan komoditas pertanian dan peternakan. (Prastowo et al., 2008).

2.1.2 Berdasarkan Struktur Pasar

Faktor penentu harga tidak hanya dipengaruhi oleh tiga proses penambahan nilai, tetapi juga dipengaruhi oleh struktur pasar. Struktur pasar ditentukan oleh

beberapa kriteria, yaitu (i) jumlah penjual yang berpartisipasi di pasar, (ii) adanya hambatan bagi penjual untuk masuk dan keluar dari pasar, dan (iii) karakteristik dari komoditas atau barang yang dijual. Secara teoritis, ada beberapa sifat struktur pasar yang berbeda dan umum ditemukan, seperti pasar monopoli, duopoli, oligopoli, persaingan monopolistik (*monopolistic competition*), dan persaingan sempurna (*perfect competition*).

Pada pasar yang bersifat monopoli, terdapat satu perusahaan atau penjual tunggal yang menguasai pasar dengan keleluasaan untuk menentukan harga. Karena tidak ada pesaing lain, perusahaan atau penjual dapat menetapkan harga sesuai keinginannya. Sebaliknya, pada pasar persaingan sempurna (*perfect competition*), perusahaan atau penjual tidak memiliki kekuatan untuk mempengaruhi harga pasar dan berperan sebagai price taker. Jika mereka mencoba menentukan harga secara mandiri, mereka berisiko kalah dalam persaingan dengan perusahaan atau penjual lain.

Pasar dengan struktur duopoli, oligopoli, dan persaingan monopolistik berada di tengah-tengah antara pasar monopoli dan persaingan sempurna dalam hal kemampuan perusahaan atau penjual untuk mempengaruhi harga. Sebagai contoh, pada level pembudidaya saat panen besar, melimpahnya jumlah komoditas pertanian yang akan dijual menyebabkan pembudidaya tidak memiliki posisi tawar untuk mempengaruhi harga dan harus menerima harga yang ada sebagai price taker, terutama jika jenis komoditas pertanian yang dijual sama dengan pembudidaya lain sehingga terjadi persaingan. Sebaliknya, ketika terdapat sedikit tengkulak dalam pasar, maka cenderung terbentuk situasi oligopoli di mana tengkulak memiliki kekuatan untuk mempengaruhi harga. Tengkulak seringkali bekerja sama untuk menentukan harga pasar dengan tujuan memperoleh keuntungan yang signifikan. (Prastowo et al., 2008).

Namun, selain tiga proses penambahan nilai ekonomi komoditas pangan dan struktur pasar, harga komoditas pangan juga dikontrol oleh pemerintah melalui kebijakan pengendalian harga pangan agar tetap terkendali. Sayangnya, kebijakan tersebut tidak akan berhasil dijalankan jika sisi produksi tidak segera diperbaiki. Hingga saat ini, perbaikan pada sisi produksi selalu dilakukan dalam bentuk bantuan atau charity, seperti pemberian pupuk, benih, traktor, pompa air, serta alat dan mesin pertanian lainnya. Namun, perbaikan semacam itu terbukti tidak efektif, ditandai dengan stagnasi produksi pangan yang menyebabkan ketergantungan pada impor. Oleh karena itu, perbaikan sisi produksi harus berfokus pada peningkatan

kesejahteraan dan kedaulatan para pembudidaya. Banyak pembudidaya yang terpaksa menjual lahan mereka kepada pengembang perumahan karena mereka tidak dapat mencukupi kebutuhan hidup dengan hasil penjualan panen mereka.

Selain itu, ada beberapa permasalahan yang memengaruhi produktivitas hasil pertanian di Indonesia, salah satunya adalah permasalahan pada aspek produksi. Beberapa isu yang terkait mencakup hal-hal berikut: (i) skala usaha pembudidaya yang masih kecil, (ii) lahan pertanian yang banyak dialihfungsikan ke non-pertanian seperti pembangunan perumahan, (iii) kerusakan infrastruktur pertanian di berbagai daerah, (iv) lemahnya sistem penyuluhan pertanian, (v) penurunan pasokan air, (vi) pertumbuhan penduduk yang tinggi, (vii) ketergantungan masyarakat pada beras, (viii) fluktuasi produksi beras, (ix) rendahnya adopsi inovasi teknologi, (x) kepemilikan lahan yang sangat kecil, (xi) kelemahan kelembagaan pembudidaya, (xii) ketergantungan pada kondisi alam dalam hal pascapanen, dan (xiii) pengaruh cuaca dan geografi setempat. (Isharyanto, 2018)

2.2 Teori Penentuan Harga Pangan dari Sisi Produksi dan Bahan Baku

Seperti yang telah dijelaskan di atas, penentuan harga jual produk dipengaruhi salah satunya oleh biaya produksi. Setelah biaya produksi telah dihitung, maka biaya tersebut perlu dijumlahkan dengan biaya penyimpanan dan biaya distribusi ke konsumen untuk menentukan harga jual produk di tingkat konsumen. Berikut penjelasan lebih lanjut mengenai biaya produksi dan komponen-komponen yang diperhitungkan dalam menentukan biaya produksi.

2.2.1 Definisi Biaya Produksi

Dikutip dari (Siregar et al., 2019), disebutkan bahwa biaya produksi adalah biaya yang dikeluarkan dalam proses mengubah bahan baku menjadi barang jadi. Biaya produksi terdiri dari tiga elemen, yaitu biaya bahan baku, biaya tenaga kerja, dan biaya overhead.

2.2.2 Komponen Harga Pokok Produksi

1. Biaya Bahan Baku

Total biaya bahan baku merujuk pada nilai keseluruhan dari bahan baku yang digunakan dalam proses produksi untuk menghasilkan barang jadi. Biaya

bahan baku memiliki peran yang sangat penting dalam perhitungan total biaya produksi barang. Sebagai contoh, dalam proses pembuatan buku, bahan baku yang digunakan mencakup kertas, tinta, lem, dan benang. Kertas menjadi bahan yang paling dominan dalam pembuatan buku, sehingga biaya kertas masuk dalam kategori biaya bahan baku. Sementara itu, bahan-bahan lain yang jumlahnya tidak terlalu signifikan akan dianggap sebagai bahan penolong dan dimasukkan ke dalam kategori biaya overhead. Siregar et al., 2019

Sebagai contoh lain, dalam usaha budidaya perikanan, diperlukan beberapa jenis biaya seperti biaya pakan, biaya vitamin dan obat-obatan, biaya distribusi, dan berbagai biaya lainnya. Biaya pakan merupakan komponen terbesar dan paling krusial, terutama ketika pembudidaya ikan melakukan budidaya dalam jumlah besar, maka dapat dikategorikan sebagai biaya bahan baku. Sementara itu, biaya lainnya akan masuk ke dalam kategori biaya penolong dan akan dikelompokkan sebagai bagian dari biaya *overhead*.

2. Biaya Tenaga Kerja

Berdasarkan sumber (Dunia et al., 2017), biaya tenaga kerja mengacu pada jumlah uang atau pembayaran yang diberikan kepada para pekerja atau karyawan yang terlibat dalam proses produksi. Biaya tenaga kerja terdiri dari dua elemen utama, yaitu biaya tenaga kerja langsung dan biaya tenaga kerja tidak langsung. Biaya tenaga kerja langsung merupakan biaya yang dapat secara khusus diidentifikasi dengan suatu operasi atau proses tertentu yang diperlukan untuk menyelesaikan produk perusahaan. Sehingga, biaya tenaga kerja langsung diatribusikan secara langsung ke komponen-komponen dari barang jadi atau produk yang dihasilkan. Biaya ini merupakan bagian dari prime cost bersama dengan biaya bahan baku dalam proses produksi.

Di sisi lain, biaya tenaga kerja tidak langsung mencakup semua biaya tenaga kerja yang tidak secara langsung terlibat dalam proses produksi dan seringkali terkait dengan departemen pendukung seperti departemen pembelian, departemen pemeliharaan, departemen pengendalian mutu, dan departemen lainnya.

3. Biaya *Overhead*

Dikutip dari (Siregar et al., 2019), biaya *overhead* melibatkan semua biaya tambahan yang diperlukan untuk memproduksi suatu barang, di samping

biaya bahan baku dan biaya tenaga kerja langsung. Beberapa contohnya termasuk biaya listrik, biaya air, biaya pajak, biaya sewa lahan, dan berbagai biaya lainnya.

Biaya produksi dapat dikelompokkan lebih lanjut menjadi dua kategori, yaitu biaya utama (*prime cost*) dan biaya konversi (*conversion cost*). Rumus untuk menghitung biaya utama adalah:

$$\text{Biaya utama} = \text{biaya bahan baku} + \text{biaya tenaga kerja langsung}$$

Sedangkan rumus untuk biaya konversi yakni:

$$\text{Biaya konversi} = \text{biaya tenaga kerja tidak langsung} + \text{biaya overhead}$$

2.3 Teori Penentuan Harga Berdasarkan Buying Power Konsumen

Keputusan tentang penetapan harga, seperti keputusan dalam elemen campuran pemasaran lainnya, harus dimulai dengan memperhatikan nilai yang diberikan kepada pelanggan. Saat pelanggan membeli produk, mereka menukarkan sesuatu yang berharga (harga) dengan manfaat dari kepemilikan atau penggunaan produk tersebut. Penetapan harga yang efektif, berdasarkan orientasi pelanggan, melibatkan pemahaman tentang seberapa besar nilai yang diberikan oleh konsumen atas manfaat yang diperoleh dari produk, dan menetapkan harga yang mencerminkan nilai tersebut.

Penetapan harga berdasarkan nilai bagi pelanggan mengacu pada penyesuaian harga berdasarkan persepsi nilai yang dimiliki oleh pembeli. Pendekatan harga berdasarkan nilai berarti pemasar tidak merancang produk dan program pemasaran terlebih dahulu, lalu menetapkan harga. Sebaliknya, harga dipertimbangkan bersama dengan seluruh elemen campuran pemasaran lainnya sebelum program pemasaran ditetapkan.(Kotler and Armstrong, 2018).

2.4 Teori Penentuan Harga Berdasarkan Pertemuan Supply-Demand

Menurut (Smith, 1776), menguraikan konsep pasokan dan permintaan sebagai "tangan tak terlihat" yang secara alami mengarahkan perekonomian. Menurut Smith, tangan tak terlihat adalah mekanisme otomatis dalam perekonomian yang menentukan harga dan distribusi. Smith menggambarkan sebuah masyarakat di

mana para pembuat roti dan tukang daging menyediakan produk yang dibutuhkan dan diinginkan oleh individu, memberikan pasokan yang memenuhi permintaan, dan mengembangkan ekonomi yang menguntungkan semua orang.

Sedangkan menurut (Nuraini, 2016), hukum permintaan pada dasarnya merupakan suatu hipotesa yang menyatakan bahwa "Jika harga suatu barang turun, maka permintaan terhadap barang tersebut akan meningkat, sebaliknya jika harga suatu barang naik, maka permintaan terhadap barang tersebut akan menurun." Asumsi ini memiliki logika yang kuat karena ketika harga suatu barang naik, konsumen cenderung mencari alternatif lain yang harganya tidak mengalami kenaikan. Jika pendapatan konsumen tetap namun harga barang naik, maka daya beli riil konsumen akan menurun, sehingga menyebabkan penurunan permintaan terhadap barang tersebut. Sebaliknya, jika harga barang turun, konsumen akan mengurangi pembelian barang lain dan lebih banyak membeli barang yang mengalami penurunan harga.

Prinsip yang mirip dengan konsep permintaan adalah hukum penawaran, yang menggambarkan hubungan antara harga barang atau jasa dengan jumlah barang atau jasa yang ditawarkan. Hukum penawaran menyatakan bahwa "Jika harga suatu barang meningkat, maka jumlah barang yang ditawarkan akan meningkat, namun sebaliknya, jika harga suatu barang menurun, maka jumlah barang yang ditawarkan akan berkurang."

Ekuilibrium pasar terjadi ketika pada harga tertentu, jumlah barang yang diminta di pasar sama dengan jumlah barang yang ditawarkan di pasar tersebut. Kondisi ekuilibrium ini dapat dijelaskan dengan berbagai cara, misalnya melalui jadwal permintaan dan penawaran, fungsi permintaan dan penawaran yang seimbang, atau melalui kurva permintaan dan penawaran.(Nuraini, 2016)

2.5 Teori Penentuan Harga yang Ditentukan Oleh Dominan Power

Strategi penetapan harga '*follow-the-leader*' adalah pendekatan persaingan di mana sebuah bisnis menyesuaikan harga dan layanannya dengan pemimpin pasar. Ini berarti perusahaan tersebut akan mengikuti harga yang ditetapkan oleh pemain terbesar di industri tersebut. Misalnya, ketika pemimpin pasar menurunkan harga barangnya, perusahaan lain akan menyesuaikan harga mereka dengan tingkat yang sama.

Strategi penetapan harga '*follow-the-leader*' dapat mendorong bisnis untuk

terus-menerus menyesuaikan harga mereka, terutama jika pemimpin pasar merespons strategi ini dengan terus-menerus menaikkan dan menurunkan harga. Namun, hal ini dapat menyebabkan terjadinya perang harga.

Perang harga terjadi ketika perusahaan dengan sengaja saling menurunkan harga untuk saling mengalahkan. Sebagai contoh, ketika satu perusahaan menurunkan harga untuk menyamai harga barang yang menjadi daya tarik utama, pemimpin pasar dapat lebih lanjut memotong harga mereka untuk mempertahankan atau mendapatkan lebih banyak pangsa pasar. Hal ini dapat berlangsung dalam jangka waktu yang lama dan berujung pada perang harga. Beberapa contoh besar perang harga telah terjadi antara Apple dan Samsung, serta Walmart dan Amazon. (Hargrave, 2019)

2.6 Konsep Lelang

Lelang adalah istilah yang mengacu pada suatu bentuk transaksi jual beli yang memiliki aturan khusus. Lelang didefinisikan sebagai proses penjualan barang yang terbuka untuk masyarakat umum, di mana penawaran harga dapat diajukan secara tertulis atau lisan, dengan harga yang bisa naik atau turun hingga mencapai harga tertinggi. Prosedur lelang dimulai dengan pengumuman mengenai barang yang akan dijual melalui lelang. Asal usul kata "lelang" sendiri berasal dari bahasa Latin "auctio," yang berarti peningkatan harga secara bertahap. Di Indonesia, praktik lelang pertama kali diatur pada tahun 1908, saat masih berada di bawah pemerintahan Belanda. Pada tahun tersebut, dasar hukum untuk lelang di Indonesia dikenal dengan istilah "Vendu Reglement" (Lembaran Negara Republik Indonesia 1908 No. 189) dan "Vendu Instructie" (Lembaran Negara Republik Indonesia 1908 No. 190). Sampai saat ini, peraturan dasar tersebut masih berlaku di Indonesia sebagai panduan untuk melaksanakan lelang (Tanaya, 2022).

2.7 Karakteristik Lelang

Dikutip dari (Kumala, 2020), lelang memiliki beberapa karakteristik sebagai berikut:

1. Sebuah transaksi melibatkan dua pihak, yaitu penjual dan pembeli.
2. Terdapat barang yang menjadi fokus transaksi.

3. Barang tersebut dijual di depan masyarakat umum.
4. Pentingnya melakukan pengumuman lelang untuk menarik minat calon pembeli melalui berbagai cara seperti pengumuman atau publikasi kepada publik.
5. Lelang dijalankan oleh seorang pejabat lelang atau dihadapan pejabat lelang sebagai perantara.
6. Harga terbentuk melalui penawaran, baik secara lisan, tertulis, atau melalui platform online dengan harga yang bisa naik atau turun.
7. Peserta lelang adalah masyarakat umum yang memenuhi syarat-syarat tertentu.
8. Penjualnya hanya ada satu, sementara calon pembelinya dapat berjumlah banyak.

2.8 Sistem Keuangan

Sistem Keuangan adalah struktur ekonomi suatu negara yang berperan dalam menyelenggarakan berbagai layanan keuangan melalui lembaga keuangan. Fungsi utama sistem keuangan adalah mengarahkan dana yang tersedia dari pihak yang menyimpan kepada pihak yang membutuhkan dana, untuk digunakan dalam pembelian barang dan jasa, serta investasi. Hal ini bertujuan untuk mendorong pertumbuhan ekonomi dan meningkatkan kualitas hidup (Soemitra, 2015).

2.9 Teori Graf

Pada penelitian ini, dibutuhkan algoritma yang dapat digunakan untuk menentukan jalur terdekat antar pembudidaya ikan, terutama pembudidaya ikan yang sedang dalam musim panen. Sebab salah satu fitur pada iterasi ketiga aplikasi ini yakni untuk menampilkan titik-titik/node yang merepresentasikan lokasi pembudidaya ikan satu sama lain pada sebuah peta/maps. Pembudidaya ikan yang tengah panen perlu mengetahui lokasi terdekatnya dengan pembudidaya lain sebab mereka perlu bertukar informasi tertentu seperti informasi harga per kilogram, informasi pakan ikan yang cocok, informasi perawatan ikan dari jamur atau penyakit, dan lain-lain. Penentuan jalur terdekat sangat penting sebab jalur tersebut dapat menghemat waktu dan ongkos (baik ongkos transportasi umum maupun

ongkos transportasi pribadi seperti bensin) sehingga perjalanan menjadi jauh lebih efektif dan cepat apabila ada kedua pembudidaya ikan yang telah setuju untuk saling bertemu. Untuk menentukan jalur terdekat antara dua titik (yang digunakan untuk merepresentasikan lokasi pembudidaya ikan), maka dibutuhkan implementasi dari teori Graf. Berikut penjelasan singkat mengenai teori Graf.

Menurut (Rosen, 2019), graf adalah struktur diskrit yang terdiri dari simpul-simpul (*vertex*) yang saling dihubungkan oleh sisi-sisinya (*edges*). Ada beberapa jenis graf yang berbeda berdasarkan jenis dan jumlah sisi yang dapat menghubungkan sepasang simpul. Persoalan/masalah di hampir setiap disiplin ilmu dapat diselesaikan dengan menggunakan model graf, seperti mengkalkulasikan jumlah kombinasi penerbangan berbeda antara dua kota pada sebuah jaringan maskapai. Secara matematis, graf dapat dirumuskan sebagai berikut:

Dikutip dari (Munir, 2003), misalkan Graf A merupakan pasangan himpunan dari (V, E) , di mana :

V merupakan himpunan tidak kosong dari simpul-simpul (*vertices* atau *nodes*) = $V_1, V_2, V_3, \dots, V_n$,

dan

E merupakan himpunan sisi (*edges*) yang menghubungkan sepasang simpul = $E_1, E_2, E_3, \dots, E_n$

Sehingga dapat ditulis secara singkat menjadi notasi

$$G = (V, E) \quad (2.3)$$

Berdasarkan notasi di atas, maka dapat disimpulkan bahwa simpul-simpul (*vertices*) pada sebuah graf tidak boleh kosong, sedangkan sisi-sisinya (*edges*) boleh kosong. Sebuah graf dimungkinkan untuk tidak memiliki sisi sama sekali, namun harus memiliki simpul minimal satu. Simpul-simpul pada graf dapat dinotasikan atau dilambangkan dengan huruf seperti $a, b, c, \dots, V, \dots, z$, atau dengan angka seperti $1, 2, 3, \dots$ atau gabungan dari keduanya. Sedangkan sisi yang menghubungkan simpul V_i dengan simpul V_j dinyatakan dengan pasangan, seperti :

$$e = (V_i, V_j) \quad (2.4)$$

Seperti yang telah dijelaskan sebelumnya, graf dapat digunakan untuk memecahkan hamper semua masalah, termasuk mencari dan menghitung jalur terdekat antara dua simpul (*vertices*) pada sebuah graf berbobot. Ada beberapa teori Graf dan algoritma yang bisa digunakan untuk menentukan jalur terdekat antara dua simpul, seperti Algoritma Dijkstra dan teori graf *local neighborhood*.

2.9.1 Teori Graf Algoritma A*

Algoritma A* tradisional adalah algoritma yang tipikal dengan heuristik. Fungsi biaya dari algoritma A* dapat didefinisikan sebagai berikut:

$$f(n) = h(n) + g(n) \quad (2.5)$$

dengan rincian n adalah titik inspeksi saat ini; $f(n)$ adalah jumlah dari fungsi penilaian; $g(n)$ adalah nilai substitusi jalur aktual dari titik awal ke titik inspeksi; dan $h(n)$ adalah estimasi nilai substitusi dari titik inspeksi ke titik akhir.

$h(n)$ dihitung menggunakan menggunakan jarak Euclidean antara titik awal dengan titik akhir:

$$h(n) = \sqrt{(X_n - X_{goal})^2 + (Y_n - Y_{goal})^2} \quad (2.6)$$

Langkah-langkah penerapan algoritma A* tradisional dapat dilihat pada pseudocode di Algoritma 1.

Namun, menurut (Ju et al., 2020) algoritma A* tradisional tidak optimal dalam mencari jalur terpendek, sehingga (Ju et al., 2020) mengembangkan algoritma A* baru yang lebih cepat 200% dalam mencari jalur dibandingkan dengan algoritma A* tradisional. Pertama-tama, bayangkan kasus sederhana dengan segmen garis lurus yang menghubungkan titik awal dan titik akhir, dengan asumsi hambatan hanya melewati satu, seperti yang ditunjukkan pada Gambar 1. Kemudian, titik C ditentukan sebagai pusat, lalu buat lingkaran dengan jari-jari r. Titik D dan E adalah titik potong lingkaran ini dan segmen garis (A-B). Pertama-tama, selesaikan persamaan segmen garis lurus (A-B) :

$$(Y - Y_{start})(X_{goal} - X_{start}) = (X - X_{start})(Y_{goal} - Y_{start}) \quad (2.7)$$

Kemudian selesaikan persamaan lingkaran dengan hambatan C yang

Algorithm 1 Algoritma A* tradisional

```

algorithmA_star(start, map, goal)
  close  $\leftarrow$  NULL
  open  $\leftarrow$  start
  G[start]  $\leftarrow$  g[start] = 0, H[start]  $\leftarrow$  h[start]
  F[start]  $\leftarrow$  H[start]
  while open $\neq\emptyset$  do
    current_Node  $\leftarrow$  Node with the minimum F value in open
    open  $\leftarrow$  open \ {current_Node}
    close  $\leftarrow$  close  $\cup$  { current_Node }
  end while
  if current_Node = goal then
    return best_path
  end if
  neighbor_Nodes  $\leftarrow$  All neighbor Nodes of current_Node
  for N  $\in$  neighbor_Nodes do
    if N  $\in$  close then
      do nothing
    else if N  $\in$  open then
      G[N_new_calculated], H, F  $\leftarrow$  calculateN'sG, H, F
      if G[N  $\in$  open] > G[N_new_calculated] then
        N'sparent  $\leftarrow$  current_Node
      end if
    else
      N'sparent  $\leftarrow$  current_Node
    end if
  end for

```

*Mencerdaskan dan
Memartabatkan Bangsa*

koordinatnya adalah $(X_{obstacles} - Y_{obstacles})$ sebagai pusat hambatan dan jari-jari r yang melewatiinya:

$$(X - X_{obstacles})^2 + (Y - Y_{obstacles})^2 = r^2 \quad (2.8)$$

Melalui dua persamaan tersebut, dapat diketahui masing-masing koordinat dari titik D dan titik E yakni $D(X_D, Y_D)$ dan $E(X_E, Y_E)$. Titik D dan titik E akan digunakan sebagai titik awal dan titik akhir, sehingga penting untuk menentukan titik mana yang akan digunakan sebagai titik awal dan titik mana yang akan digunakan sebagai titik akhir, lalu membuat penilaian berikut:

$$(X_E - X_D) = L \quad (2.9)$$

Jika nilai L lebih besar dari 0, maka pilihlah titik D sebagai titik awal dan titik E sebagai titik akhir. Sebaliknya, jika nilai L lebih kecil dari 0, maka pilihlah titik E sebagai titik awal dan titik D sebagai titik akhir. Berdasarkan Gambar 1, jelas bahwa D adalah titik awal dan E adalah titik akhir, kemudian titik awal lokal D dan titik akhir lokal E dianggap sebagai kondisi yang telah diketahui, lalu dengan mengganti ke algoritma A* tradisional pada bagian sebelumnya, sebuah jalur akan dihasilkan dengan D sebagai titik awal, E adalah jalur optimal dari titik akhir, yaitu busur hitam DGE.

Kemudian titik jalur setelah menyatukan jalur ini dengan jalur yang sebelumnya dihasilkan adalah [A,D,G,E,B,J], sebuah garis yang diperkirakan hampir lurus telah diciptakan.

Prinsip yang sama dengan contoh yang diberikan di atas, penerapan algoritma A* yang telah ditingkatkan dapat dilihat pada Algoritma 2.

2.9.2 Teori Graf tentang Ketetanggaan Lokal (Local Neighborhood)

Dikutip dari (Hacid and Zighed, 2005), *Local Neighborhood* atau grafik kedekatan, adalah struktur geometris yang menggunakan konsep ketetanggaan untuk menentukan titik terdekat dengan titik yang diberikan oleh titik permintaan. Grafik kedekatan ini sangat didasarkan pada jarak antar titik. Berikut adalah notasi dari grafik kedekatan :

Misalkan Ω adalah himpunan titik dalam ruang multidimensi R^d . Graf $G(\Omega, \Phi)$ disusun oleh himpunan titik Ω dan himpunan sisi Φ . Kemudian, untuk setiap grafik relasi biner dapat diasosiasikan pada Ω , di mana dua titik $(p, q) \in \Omega^2$

Algorithm 2 Algoritma A* yang telah ditingkatkan

```

algorithm improved_Astar(mulai, map, tujuan)
    Line  $\leftarrow$  a line connecting the start and end goals
    circle_sum, center, local_coordinate  $\leftarrow \emptyset$ 
    for each: obs  $\in$  obstacle
        center  $\leftarrow$  center  $\cup$  ispass_line(obs, Line)
    for each: c  $\in$  center
        local_coordinate  $\leftarrow$  local_coordinate  $\cup$  solve_line_circle(c, r, Line)
        for (local_start, local_goal)  $\in$  (local_coordinate) do
            if  $\|$  local_goal(i) - local_start(i + 1)  $\| \leq \varepsilon$  then
                [local_start(i), local_goal(i + 1)]  $\leftarrow$  [local_start(i + 1), local_(i + 1)]
            end if
        end for
        for (local_start, local_goal)  $\in$  (local_coordinate) do
            path  $\leftarrow$  path  $\cup$  A_star(start, map, goal)
        end for
        Line  $\leftarrow$  Line  $\setminus$  {circle_sum}
        path_final  $\leftarrow$  Line  $\cup$  path
        ispass_line(obs, Line)
        if then obs  $\in$  Line
            return obs
        end if
        solve_line_circle(c, r, Line)
        circle  $\leftarrow$  a circle with c as the center and r as the radius
        circle_sum  $\leftarrow$  circle_sum  $\cup$  circle
        D, E  $\in$  (Line  $\cap$  circle)
    return D, E

```

*Mencerdaskan dan
Memartabatkan Bangsa*

berada di relasi biner jika dan hanya jika pasangan $(p, q) \in \Phi$. Dengan cara lain, (p, q) adalah relasi biner (R) jika dan hanya jika mereka terhubung langsung dalam graf G . Dari sini, lingkungan $N(p)$ dari titik p dalam grafik G dapat dipertimbangkan sebagai sub-graf yang berisi titik p dan semua titik yang langsung terhubung dengannya.

Beberapa kemungkinan diusulkan untuk membangun grafik lingkungan. Beberapa di antaranya yakni triangulasi Delaunay, Graf Lingkungan Relatif, Graf Gabriel, dan pohon merentang minimum. Pada tulisan ini, penulis hanya mengutip penjelasan terkait Graf Lingkungan Relatif.

Dalam grafik ketetanggaan relatif $GRNG(\Omega, \Phi)$, dua titik $(p, q) \in \Omega^2$ dianggap bertetangga jika mereka melakukan pemeriksaan pada properti ketetanggaan yang relatif yang ditentukan selanjutnya.

Misalkan $H(p, q)$ adalah hiperbola dengan jari-jari $d(p, q)$ dan berpusat pada p , dan misalkan $H(q, p)$ adalah hiperbola dengan jari-jari $d(q, p)$ dan berpusat di q .

$d(p, q)$ dan $d(q, p)$ adalah ukuran perbedaan antara dua titik p dan q . $d(p, q) = d(q, p)$.

Maka, p dan q bertetangga jika dan hanya jika *lune* (bentuk bulan sabit yang dibentuk oleh dua lingkaran berpotongan) $A(p, q)$ yang dibentuk oleh perpotongan dua hiperbola $H(p, q)$ dan $H(q, p)$ kosong. Secara formal:

$$A(p, q) = H(p, q) \cap H(q, p) \text{ Then } (p, q) \in \varphi \text{ if } A(p, q) \cap \Omega = \emptyset \quad (2.10)$$

Lebih lanjut, dalam penelitian (Hacid and Zighed, 2005), dianggap bahwa tugas untuk memperbarui graf ketetanggaan lokal melibatkan lokasi titik yang dimasukkan atau titik kueri (atau dalam kasus lain, titik yang dihapus), serta titik-titik yang dapat dipengaruhi oleh pembaruan tersebut. Proses ini mencakup dua tahap utama. Pada tahap pertama, menentukan area ruang optimal yang dapat mencakup jumlah maksimum titik yang mungkin paling dekat dengan titik kueri (titik yang sedang dicari dalam ruang multidimensi). Tahap kedua adalah menyaring item yang telah ditemukan sebelumnya. Penyaringan ini dilakukan untuk mengidentifikasi titik-titik yang sebenarnya paling dekat dengan titik kueri dengan menerapkan properti ketetanggaan yang sesuai. Tahap terakhir ini menghasilkan pembaruan yang efektif dari hubungan ketetanggaan antara titik-titik yang relevan. Berikut adalah ringkasan langkah-langkah metode tersebut:

1. Susun titik q di dalam ruang multidimensi Rd;
2. Cari tetangga terdekat pertama dari q (disebut p);
3. Tetapkan area pencarian yang dimulai dari p;
4. Pindai area yang terkait dan perbarui koneksi antara item yang ada di sana;
5. Ambil tetangga-tetangga yang benar-benar dekat dengan q dan lihat apakah ada pengecualian.

Tahap utama dalam metode ini adalah penentuan area pencarian. (Hacid and Zighed, 2005) memanfaatkan struktur umum graf ketetanggaan untuk menentukan sinar hiperbola. (Hacid and Zighed, 2005) berfokus, terutama, pada konsep tetangga terdekat dan tetangga terjauh. Jadi, dua pengamatan terkait dengan dua konsep ini dapat disimpulkan menjadi:

1. Himpunan tetangga dari tetangga terdekat adalah kandidat tetangga potensial untuk titik kueri q. Dari sana, kita dapat menyimpulkan bahwa:
2. Semua tetangga suatu titik juga menjadi kandidat untuk tetangga suatu titik di mana titik tersebut adalah tetangganya.

Untuk membentuk himpunan-himpunan tersebut, langkah awalnya adalah menentukan suatu hiperbola yang memaksimalkan peluang untuk mencakup tetangga-tetangga dari titik yang dimasukkan. Setelah itu, dilakukan penyaringan terhadap item-item yang ditemukan untuk menemukan tetangga-tetangga yang valid sebagian.

Dalam langkah pertama, ditentukan jari-jari hiperbola (SR) yang memaksimalkan kemungkinan untuk mencakup tetangga-tetangga dari titik yang dimasukkan. Jari-jari ini berhubungan dengan jari-jari bola yang mencakup semua tetangga dari tetangga terdekat pertama terhadap titik kueri. Untuk menghitung jari-jari tersebut, (Hacid and Zighed, 2005) memperhatikan jarak antara titik yang dimasukkan dan tetangga terdekatnya (d_1), serta jarak antara tetangga terdekat dan tetangga terjauh (d_2). Dengan kata lain, jika q adalah titik kueri dan p adalah tetangga terdekat dari q dengan jarak d_1 , dan X adalah tetangga terjauh dari p dengan jarak d_2 , maka:

$$SR = d_1 + d_2 + \epsilon \quad (2.11)$$

dimana :

1. SR merupakan nilai dari jari-jari hiperbola SR.
2. d_1 merupakan jarak antara titik yang dimasukkan dan tetangga terdekatnya.
3. d_2 merupakan jarak antara tetangga terdekat dan tetangga terjauh.
4. ϵ merupakan parameter yang dapat diatur, baik berdasarkan karakteristik data (seperti tingkat dispersi) atau pengetahuan dalam bidang tersebut. Untuk menghindari dampak dimensi yang tinggi, kami secara eksperimental menetapkan nilai parameter ini menjadi 1.

2.10 Maps API

Sesuai yang telah dijelaskan sebelumnya, pada aplikasi iterasi ketiga ini akan ada fitur di mana pembudidaya ikan dapat melihat lokasi masing-masing pada sebuah peta untuk saling mendapatkan informasi bila pembudidaya-pembudidaya tersebut sedang dalam masa panen ikan. Untuk dapat menampilkan peta di aplikasi smartphone membutuhkan bantuan Maps API. Salah satu Maps API terkenal dan paling sering digunakan yakni Google Maps, namun ketika penelitian ini ditulis, layanan API Google Maps telah berbayar dengan tagihan yang agak mahal, sehingga penulis memutuskan untuk melakukan pengkajian antara ketiga alternatif Maps API yakni OpenstreetMap, Mapbox dan HERE Location Services.

Berikut merupakan tabel perbandingan ketersediaan fitur dari ketiga alternatif Maps API tersebut:

Tabel 2.1: Perbedaan Fitur pada masing-masing Maps API

No	Fitur	OpenstreetMap	Mapbox	HERE
1	Static Map	Ada	Ada	Ada
2	Routing	Tidak ada	Ada	Ada
3	Geocoding	Tidak ada	Ada	Ada
4	Marker	Ada	Ada	Ada

No	Fitur	OpenstreetMap	Mapbox	HERE
5	Custom Maps	Tidak ada	Ada	Tidak Ada
6	Search (Lokasi, POI, dll)	Tidak ada	Ada	Ada
7	Data <i>Traffic Real-Time</i>	Tidak ada	Ada	Ada
8	<i>Indoor Mapping</i>	Tidak ada	Ada	Ada
9	<i>Polyline dan Polygon</i>	Ada	Ada	Ada
10	Street View	Tidak ada	Tidak ada	Tidak ada

Berikut tabel perbandingan harga untuk fitur utama dari ketiga alternatif Maps API per 1000 *request*:

Tabel 2.3: Perbedaan Harga pada masing-masing Maps API

No	Fitur	OpenstreetMap	Mapbox	HERE
1	Direction API	Free	Free until 100.000 request/month, then it cost \$2.00 for the next 500.000 request.	Free until 30.000 request/month, then it cost \$0.75.

No	Fitur	OpenstreetMap	Mapbox	HERE
2	Static Map	Free	Free until 25.000 Active Users/month, then it cost \$4.00 for the next 100.000 Active Users.	Free until 30.000 request/month, then it cost \$0.075.
3	Geocoding	Free	\$5.00 until 500.000 Request/month, then it cost \$4.00 for the next 500.000 Request.	Free until 30.000 request/month, then it cost \$0.75.
4	Search Location	Free	Free until 100.000 request/month, then it cost \$0.75 for the next 500.000 request.	Free until 5.000 request/month, then it cost \$2.50.
5	Autosuggest / Autofilled	Not Available	Free until 1000 request/month, then it cost \$12.50 for the next 25.000 request.	Free until 5.000 request/month, then it cost \$2.50.

Mencerdaskan dan Memartabatkan Bangsa

Selain menampilkan peta, aplikasi iterasi ketiga ini juga akan menampilkan fitur Direction API, yakni fitur yang dapat menunjukkan arah dari lokasi pengguna

ke lokasi yang hendak dituju. Dalam konteks penelitian ini, maka fitur tersebut menunjukkan arah dari lokasi satu pembudidaya menuju pembudidaya lain yang hendak dituju. Sayangnya, fitur Direction API di Google Maps berbayar, dan untuk mencari alternatif maka penulis mengkaji fitur Direction API yang disediakan dari masing-masing alternatif Maps API, berikut hasil pengkajian implementasi Direction API:

1. OpenstreetMap Untuk dapat menggunakan OpenstreetMap di aplikasi Flutter, maka membutuhkan bantuan package *flutter_map* sebab OpenstreetMap hanya menyediakan API untuk mengedit database mereka saja, sehingga untuk menampilkan peta, menggambar polyline dan lain-lain harus menggunakan bantuan package pihak ketiga. Package *flutter_map* merupakan package yang dapat menampilkan peta dari berbagai sumber url seperti Google Maps, OpenStreetMaps, Mapbox, dan lain-lain. Package ini juga dilengkapi dengan fitur draw polyline, yakni fitur yang dapat menghubungkan lokasi A dengan lokasi B dengan garis lurus. Namun, package ini tidak memiliki fitur Direction API bawaan sehingga perlu menggunakan API lain agar dapat menggunakan fitur tersebut. Salah satu Direction API yang gratis dan banyak digunakan yakni Direction API milik Openrouteservice. Openrouteservice memiliki endpoint Directions sebagai berikut:

https://api.openrouteservice.org/v2/directions/profile?api_key=api_key&start=start_coordinates&end=end_coordinates

API tersebut memiliki satu *path parameter* yakni *profile* dan tiga *query parameters* yakni *api_key*, *start* dan *end*. *Path parameter profile* merupakan *path parameter* yang wajib diisi (tidak boleh kosong atau *null*) dengan jenis moda transportasi yang digunakan oleh pengguna. Ada sembilan jenis moda transportasi yang dapat digunakan untuk mengisi nilai *path parameter profile* tersebut, yakni:

- (a) Mobil: *driving-car*
- (b) HGV(*Heavy Goods Vehicle*; truk besar): *driving-hgv*
- (c) Sepeda: *cycling-regular*
- (d) Sepeda balap: *cycling-road*

- (e) Sepeda gunung: *cycling-mountain*
- (f) Sepeda elektrik: *cycling-electring*
- (g) Jalan kaki: *foot-walking*
- (h) Mendaki: *foot-hiking*
- (i) Kursi roda: *wheelchair*

Query parameter api_key wajib diisi dengan *key* yang unik dari setiap pengguna API tersebut. *Key* tersebut penting dan wajib diisi sebab bila dikosongkan maka API tersebut akan mengembalikan kode 401(*Unauthorized*). *Key* tersebut bisa didapatkan secara gratis dan mudah, yakni cukup dengan mendaftarkan akun baru di situs <https://openrouteservice.org/dev> kemudian klik tombol *Create Token* di menu Dev Dashboard. Setelah *key* tersebut berhasil dibuat, maka *key* tersebut harus digunakan untuk mengisi nilai dari *query parameter api_key* agar API tersebut tidak lagi mengembalikan kode 401 dan mengembalikan objek JSON yang sebenarnya.

Query parameter start dan *end* harus diisi dengan nilai *longitude* dan *latitude* dari lokasi keberangkatan dan lokasi tujuan yang dipisahkan dengan tanda baca koma. Contoh:

```
start=106.87919451247053,-6.194158605350978 (UNJ Kampus A)
end=106.88840350424874, -6.193145583778117 (UNJ Kampus B)
```

Berikut contoh penggunaan url Direction API dengan moda transportasi mobil, lokasi keberangkatan Universitas Negeri Jakarta Kampus A dan lokasi tujuan Universitas Negeri Jakarta Kampus B:

https :

```
//api.openrouteservice.org/v2/directions/driving-car?api_key =
5b3ce3597851110001cf6248102d5e41d98b43d78f243e0b52066400&start =
106.87919451247053, -6.194158605350978&end =
106.88840350424874, -6.193145583778117
```

Object JSON yang dikembalikan oleh API tersebut dapat dilihat pada Lampiran I poin 1.1.

Untuk dapat memanggil API tersebut di aplikasi Flutter, maka diperlukan sebuah function. Berikut merupakan contoh function Direction API:

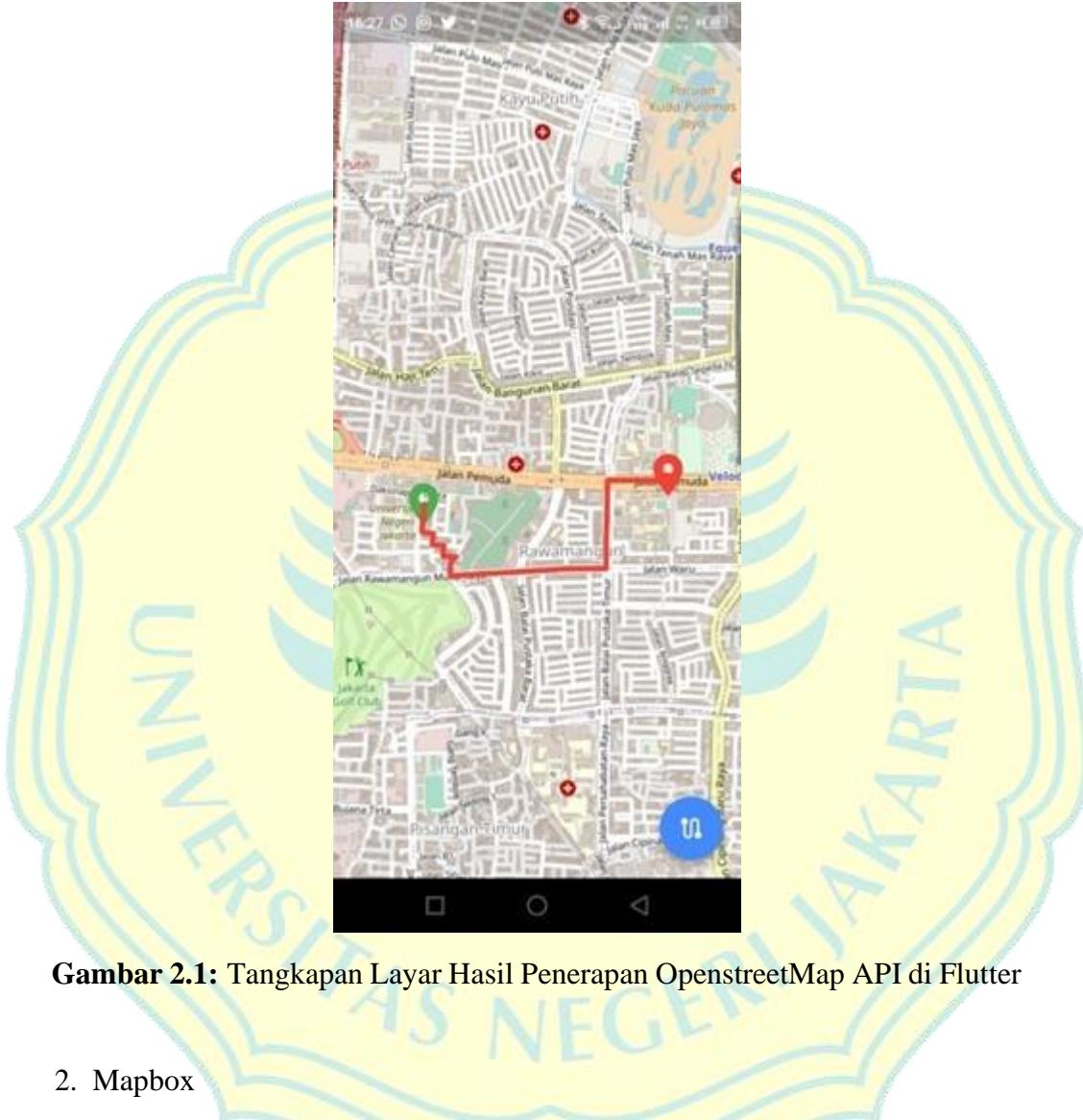
```
Future<Response> getDirection() {
    return dio.get(
        'directions/driving-car?api_key=5
b3ce359785110001cf6248102d5e41d98b43d78f243e0b52066400&
start=106.87919451247053,-6.194158605350978&end
=106.88840350424874, -6.193145583778117',
    );
}
```

Setelah function tersebut dipanggil, maka diperlukan sebuah variabel dengan tipe data `List<Latlng>` (`Latlng` merupakan singkatan dari Latitude dan Longitude) untuk menyimpan nilai coordinates dari objek JSON di atas. Langkah ini penting sebab variabel bertipe data `List<Latlng>` tersebut akan digunakan sebagai nilai parameter pada widget `Polyline` yang disediakan oleh package `flutter_map`. Widget tersebut akan menggambar garis penghubung antara lokasi A dengan lokasi B sesuai dengan nilai variabel bertipe data `List<Latlng>` yang diberikan. Berikut merupakan contoh penerapan widget `Polyline` yang telah diisi dengan nilai dari variabel bernama `points` dan bertipe data `List<Latlng>`:

```
PolylineLayer(
    polylineCulling: false,
    polylines: [
        Polyline(
            points: points!, color: Colors.red,
            strokeWidth: 5),
    ],
),
```

Berikut merupakan hasil tangkapan layar penerapan package, function dan widget yang telah dijabarkan di atas:

*Mencerdaskan dan
Memartabatkan Bangsa*



Gambar 2.1: Tangkapan Layar Hasil Penerapan OpenstreetMap API di Flutter

2. Mapbox

Berbeda dengan OpenStreetMap yang memerlukan bantuan Openrouteservice agar dapat menggunakan Direction API, Mapbox telah menyediakan fitur Direction API bawaan berbayar dengan biaya yang telah dipaparkan pada tabel di atas. Sehingga, pada penerapannya menjadi jauh lebih mudah dan sederhana. Berikut merupakan endpoint url Direction API dari Mapbox:

```
https://api.mapbox.com/directions/v5/{profile}/{coordinates}?access_token={access_token}
```

API tersebut memiliki dua *path parameters* yakni *profile* dan *coordinates*, serta memiliki satu *query parameter* yakni *access_token*. Sama seperti Direction API dari Openrouteservice, *path parameter profile* wajib diisi dengan jenis moda transportasi yang akan digunakan. Namun, jenis moda transportasi dari Mapbox hanya ada 4 jenis saja, yakni:

- (a) Berkendara di jalan raya: *mapbox/driving-traffic*
- (b) Berkendara: *mapbox/driving*
- (c) Berjalan kaki: *mapbox/walking*
- (d) Bersepeda: *mapbox/cycling*

Kemudian, untuk *path parameter coordinates*, diisi dengan nilai *longitude* dan *latitude* dari lokasi keberangkatan dan lokasi tujuan. Berikut merupakan contoh dari nilai *path parameter coordinates* dengan lokasi keberangkatan Universitas Negeri Jakarta Kampus A dan lokasi tujuan Universitas Negeri Jakarta Kampus B:

**106.87919451247053,-6.194158605350978;106.88840350424874,-
6.193145583778117**

Longitude dan *latitude* masing-masing lokasi dipisahkan dengan tanda baca koma (,), kemudian koordinat lokasi keberangkatan dan lokasi tujuan dipisahkan dengan tanda titik koma (;).

Query parameter access_token didapat dengan cara membuat akun Mapbox baru, kemudian klik tombol "Create a token" pada situs <https://account.mapbox.com/>. Namun, untuk membuat *access_token*, diperlukan *Credit* atau *Debit Card* karena Mapbox menerapkan sistem *Pay as You Go*, yakni sistem yang akan menagih pengguna hanya pada servis yang digunakan saja. Walaupun Mapbox menyediakan servis gratis dengan limit yang telah dipaparkan pada tabel di atas, namun pada proses pendaftaran tetap membutuhkan *Credit* atau *Debit Card* sebab bila pengguna telah melebihi limit maka akan langsung ditagihkan.

Berikut merupakan contoh penggunaan endpoint Direction API dari Mapbox, dengan lokasi keberangkatan Universitas Negeri Jakarta Kampus A dan lokasi tujuan Universitas Negeri Jakarta Kampus B dengan menggunakan moda transportasi berkendara di lalu lintas:

```
https://api.mapbox.com/directions/v5/mapbox/
driving-traffic
/106.87919451247053,-6.194158605350978;106.88840350424874,
-6.193145583778117?access_token=sk.
eyJ1IjoicGVtZWx1a3NlbmphyIiwiYSI6ImNsZmYycGhoczJ2NG00N
HBjaTJobjFubTEifQ.jn6QPpCbc7psplmigght2g&geometries=geojson
```

Objek JSON yang dikembalikan oleh endpoint tersebut dapat dilihat pada Lampiran I poin 1.2.

3. HERE Location Services

Sama seperti Mapbox, Here Location Services juga menyediakan endpoint Routing API (di Here Location Services, mereka menggunakan istilah “Routing API” bukan Direction API, namun memiliki fungsionalitas yang sama) milik mereka sendiri, sehingga tidak perlu menggunakan API pihak ketiga seperti Openrouteservice. Berikut merupakan endpoint url Routing API dari Here Location Services:

```
https://router.hereapi.com/v8/routes?
transportMode={transportMode}&origin={origin}&destination={destination};
sideOfStreetHint={sideOfStreetHint}&return=polyline&apikey={apikey}
```

API tersebut memiliki 4 *query parameter* yakni *transportMode*, *origin*, *destination*, dan *apikey*. *Query parameter transportMode* wajib diisi dengan jenis moda transportasi yang akan digunakan. Untuk saat ini, hanya tersedia 5 jenis moda transportasi yang dapat digunakan untuk mengisi nilai *transportMode*, yakni:

- (a) Jalan kaki: *pedestrian*
- (b) Mobil: *car*
- (c) Truk: *truck*
- (d) Sepeda: *bicycle*
- (e) Skuter: *scooter*

Query parameter origin dan *destination* merupakan koordinat *latitude* dan *longitude* dari lokasi keberangkatan dan lokasi tujuan. Pada *query parameter destination*, terdapat satu parameter lagi yakni parameter *sideOfStreetHint*. Parameter ini berguna untuk mengantarkan pengguna pada sisi jalan yang benar di lokasi tujuan.



Gambar 2.2: Tangkapan Layar Google Maps untuk Memahami *Query Parameter sideOfStreetHint*

Bila lokasi tujuan memiliki jalur dua arah seperti yang tertera di gambar, penting bagi aplikasi peta untuk dapat mengarahkan pengguna menuju sisi jalan yang benar dekat dengan lokasi tujuan. Sebab, bila aplikasi mengarahkan pengguna pada sisi yang berseberangan dengan lokasi tujuan, tentu ini akan merepotkan sebab pengguna perlu mencari jalur memutar agar dapat berada pada sisi jalan yang benar dengan lokasi tujuan. Parameter ini sudah diterapkan secara default, sehingga bila tidak dimasukkan ke dalam parameter endpoint nya pun hasil yang diberikan tetap berada di sisi jalan yang sama dengan lokasi tujuan. Namun, parameter ini hanya memberikan hasil di sisi jalan yang benar hanya ketika jalur tersebut tidak memungkinkan untuk menyebrang atau memutar balik. Sehingga, bila lokasi tujuan berada di jalur satu arah, namun jalur tersebut cukup lebar, maka pengguna tetap harus “menyebrang” untuk mencapai lokasi tujuan.

Agar pengguna selalu diarahkan ke sisi jalan yang sama dengan lokasi tujuan,

maka perlu menambahkan satu parameter lagi yakni *matchSideofStreet* yang diisi dengan nilai "always" (selalu), sehingga *endpoint url* nya menjadi seperti berikut:

```
https://router.hereapi.com/v8/routes?
transportMode={transportMode}&origin{origin}&destination={destination};sideOfStreetHint={sideOfStreetHint};
matchSideOfStreet=always&return=polyline&apikey={apikey}
```

Query parameter apikey didapat dengan cara mendaftarkan akun pada situs *Here Location Services*, kemudian klik tombol "*Create project*", kemudian setelah proyek berhasil dibuat, maka pengguna dapat membuat "key" untuk "*authorization*" dan "*apikey*". Untuk membuat "*apikey*" tidak memerlukan "*Credit*" atau "*Debit Card*", namun bila ingin menggunakan Flutter SDK-nya maka perlu menggunakan *Credit* atau *Debit Card*.

Berikut merupakan contoh endpoint Routing API dengan lokasi keberangkatan Universitas Negeri Jakarta Kampus A dan lokasi tujuan Universitas Negeri Jakarta Kampus B dengan moda transportasi mobil dan akan diarahkan pada sisi jalan yang sama dengan lokasi tujuan:

```
https://router.hereapi.com/v8/routes?
transportMode=car&origin
=-6.194158605350978,106.87919451247053&destination
=-6.193145583778117,106.88840350424874;sideOfStreetHint
=-6.193145583778117,106.88840350424874;matchSideOfStreet=
always&return=polyline&apikey=D-6
jNZpFtcZ5ZDv0s5w6EdZnVx2a1xsr58BRM-NKrLo
```

Objek JSON yang dikembalikan oleh endpoint tersebut dapat dilihat pada Lampiran I poin 1.3.

2.11 Tingkat Keakuratan Maps API

Sebagai pilihan alternatif dari Google Maps, tentu saja masing-masing Maps API yang telah dibahas sebelumnya memiliki tingkat akurasi yang berbeda-beda

karena memiliki sumber data yang berbeda-beda pula. OpenStreetMaps mengandalkan data yang diberikan oleh komunitas, sehingga tingkat akurasinya sangat bergantung pada data yang dikontribusikan oleh komunitas. Mapbox dan HERE Location Services walau cukup popular karena tingkat akurasinya di luar negeri, namun tingkat akurasinya belum tentu sama tinggi nya di Indonesia sebab sumber datanya tidak diperbarui secara cepat seperti Google yang memiliki tim untuk memetakan Indonesia. Penulis telah melakukan pengkajian tingkat akurasi mayoritas di daerah Kota Bekasi (terutama Bekasi Barat) sebab daerah tersebut merupakan daerah yang familiar dan sering dilewati oleh penulis sehingga penulis bisa membandingkan tingkat akurasi antar Maps API. Berikut merupakan beberapa hasil pengkajian yang dilakukan oleh penulis:

1. OpenStreetMaps

(a) Stasiun Bekasi

Stasiun Bekasi sudah berdiri sejak tahun 1887, sehingga sangat wajar apabila stasiun ini sudah ditampilkan pada setiap Maps API alternatif dari Google Maps, salah satunya yakni OpenStreetMaps. Namun sayangnya, OpenStreetMaps hanya menandai Stasiun Bekasi sebagai satu kotak berwarna biru saja, dan tidak ada pembeda pintu masuk stasiun. Stasiun Bekasi memiliki dua pintu masuk, yakni pintu masuk Perjuangan dan pintu masuk Juanda. Sayangnya, hal ini tidak ditampilkan pada peta OpenStreetMaps sehingga cukup membingungkan bagi pengguna yang baru pertama kali ke Stasiun Bekasi, sebab mereka pasti akan mencoba ke Jl. Pemuda untuk memasuki stasiun terlepas dari arah mana mereka datang, sehingga apabila mereka datang dari arah Jl. Ir. H. Juanda pun mereka akan mencoba memutar ke Jl. Pemuda padahal ada satu pintu masuk di Jl. Ir. H. Juanda. Berikut penulis lampirkan tampilan layar peta OpenStreetMaps dengan lokasi Stasiun Bekasi:

*Mencerdaskan dan
Memartabatkan Bangsa*

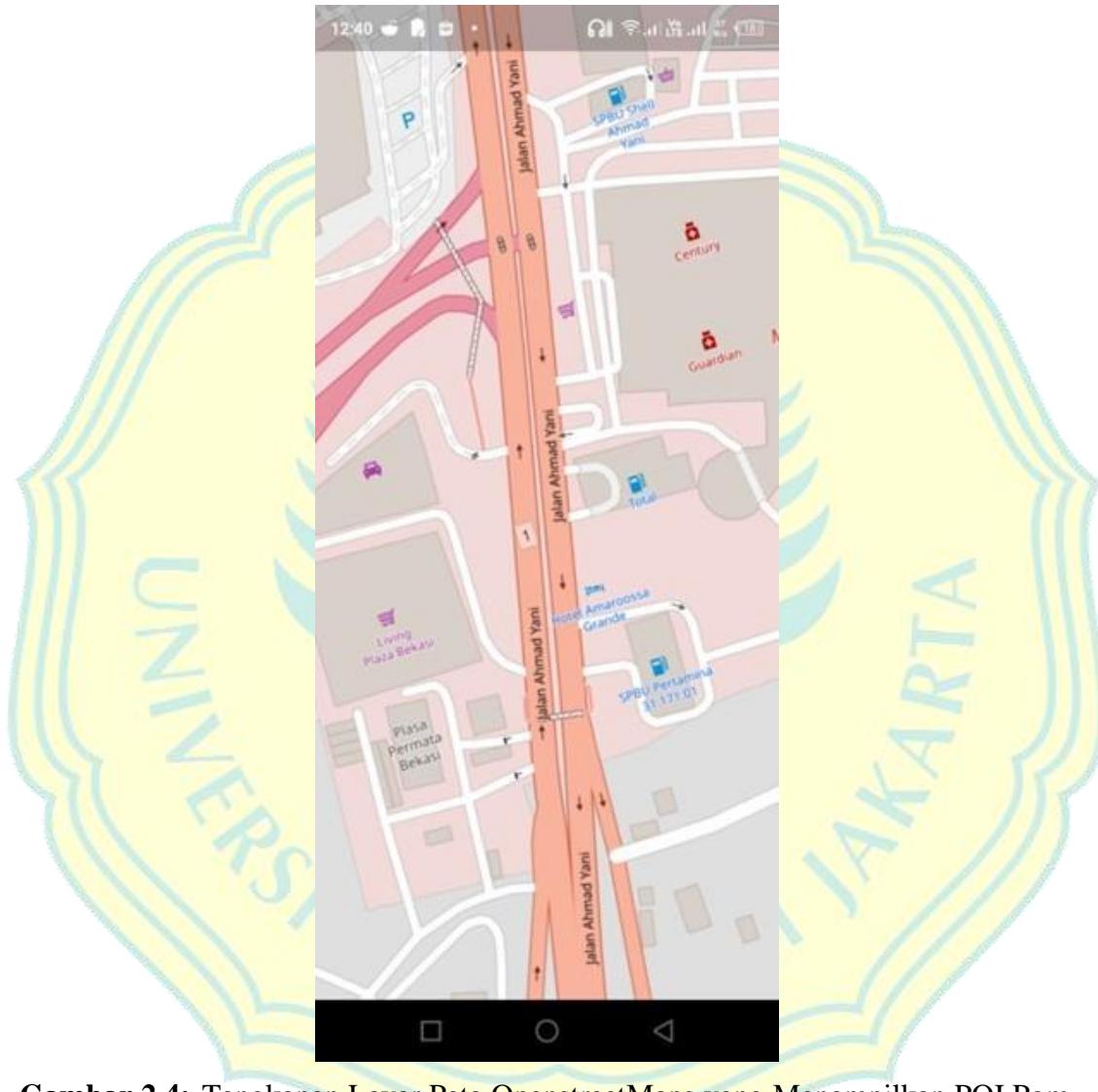


Gambar 2.3: Tangkapan Layar Peta OpenstreetMaps dengan lokasi Stasiun Bekasi

(b) Pom Bensin Total

Pom Bensin Total merupakan salah satu SPBU swasta yang cukup terkenal di Indonesia. Sayangnya, pada akhir tahun 2020, Pom Bensin Total telah menutup seluruh SPBU nya secara total sehingga (ketika skripsi ini ditulis) sudah tidak ada lagi Pom Bensin Total di Indonesia. Di Kota Bekasi, lokasi yang sebelumnya ditempati oleh Pom Bensin Total (dekat dengan Hotel Amarossa Grande) kini telah ditempati oleh SPBU swasta baru yakni Vivo. Namun, di OpenStreetMaps, POI yang ditampilkan masih Pom Bensin Total bukan Pom Bensin Vivo, sehingga masih tidak akurat. Berikut penulis lampirkan tampilan layar yang

menampilkan POI Pom Bensin Vivo di peta OpenStreetMaps :



Gambar 2.4: Tangkapan Layar Peta OpenstreetMaps yang Menampilkan POI Pom Bensin Total

(c) Kota Cinema Mall Wisma Asri Bekasi

Di persimpangan Rumah Sakit Tiara Bekasi, bila pengendara memilih untuk mengambil jalur sebelah kiri, maka pengendara akan melewati sebuah gedung Kota Cinema Mall Wisma Asri Bekasi, kemudian tak lama akan melintasi jembatan besi untuk menyeberangi sungai. Namun, Kota Cinema Mall Wisma Asri Bekasi baru didirikan pada tahun 2017. Sebelumnya, di gedung yang sama, berdirilah Giant Hypermarket, yang kemudian tutup karena mengalami kebangkrutan dan digantikan oleh

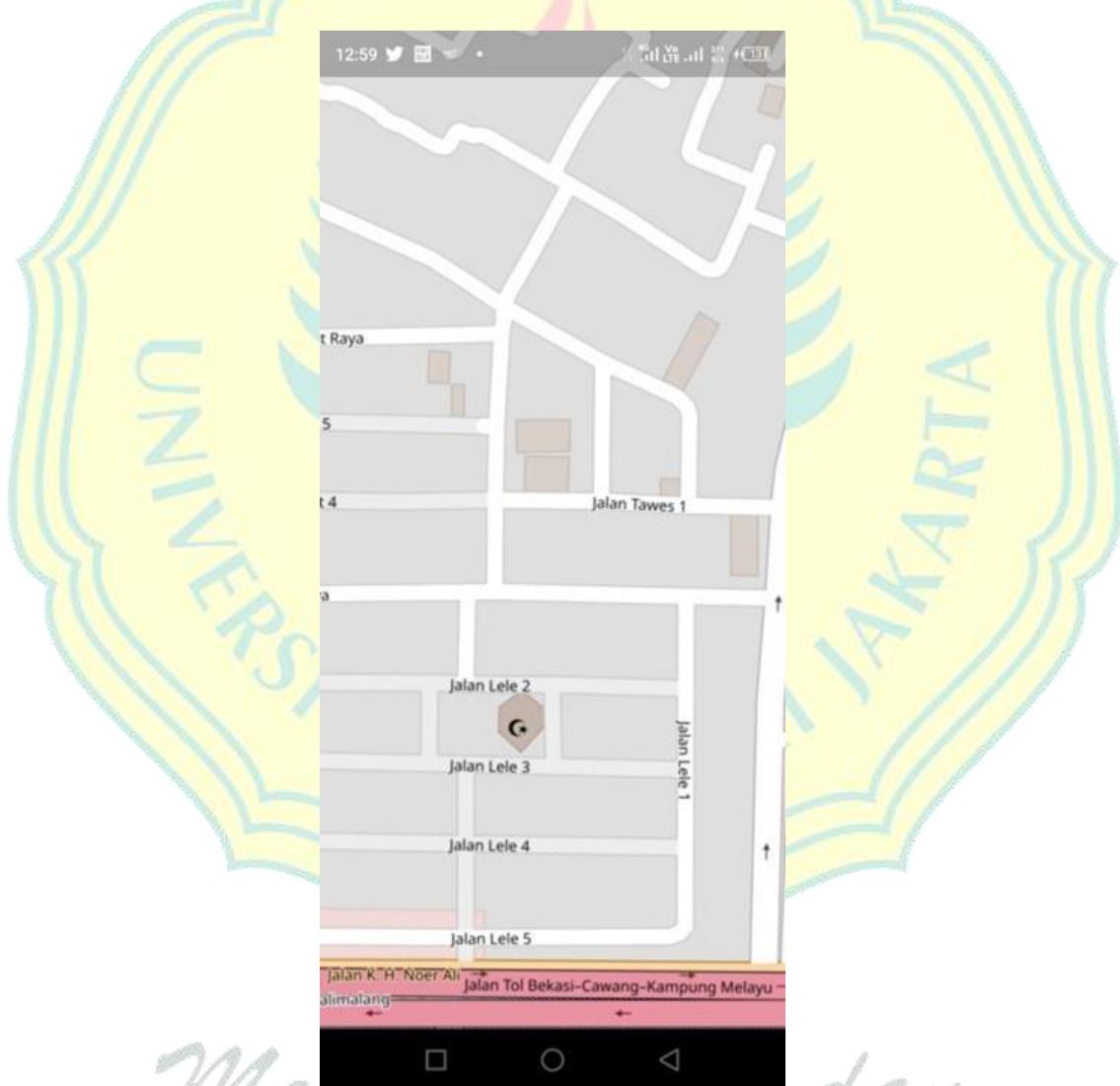
Kota Cinema Mall Bekasi. Ketika penulis melakukan pengkajian, penulis menemukan bahwa Kota Cinema Mall Bekasi telah ditambahkan sebagai POI menggantikan Giant Hypermarket di peta OpenStreetMaps. Temuan ini membuktikan bahwa setidaknya peta OpenStreetMaps telah diperbarui hingga tahun 2017, dan belum diperbarui sejak pertengahan tahun 2020. Sebab, Pom Bensin Total yang telah tutup sejak akhir tahun 2020 masih ditampilkan sebagai titik POI di peta. Berikut penulis lampirkan tampilan layar peta OpenStreetMaps dengan titik POI Kota Cinema Mall Bekasi:



Gambar 2.5: Tangkapan Layar Peta OpenstreetMaps yang Menampilkan POI Kota Cinema Mall Wisma Asri Bekasi

(d) Detail Tampilan Pemukiman Warga

Setiap Maps API memiliki keunikannya tersendiri ketika menampilkan lingkungan pemukiman warga. Di OpenStreetMaps, lingkungan pemukiman warga hanya ditampilkan nama jalannya saja dan tidak menampilkan nomor rumahnya. Berikut tampilan tangkapan layar untuk pemukiman warga di Jl. Lele, Kayuringin Jaya, Bekasi:



Gambar 2.6: Tangkapan Layar Peta OpenstreetMaps yang Menampilkan Pemukiman Warga di Jl. Lele, Kayuringin Jaya, Bekasi

2. Mapbox

(a) Stasiun Bekasi

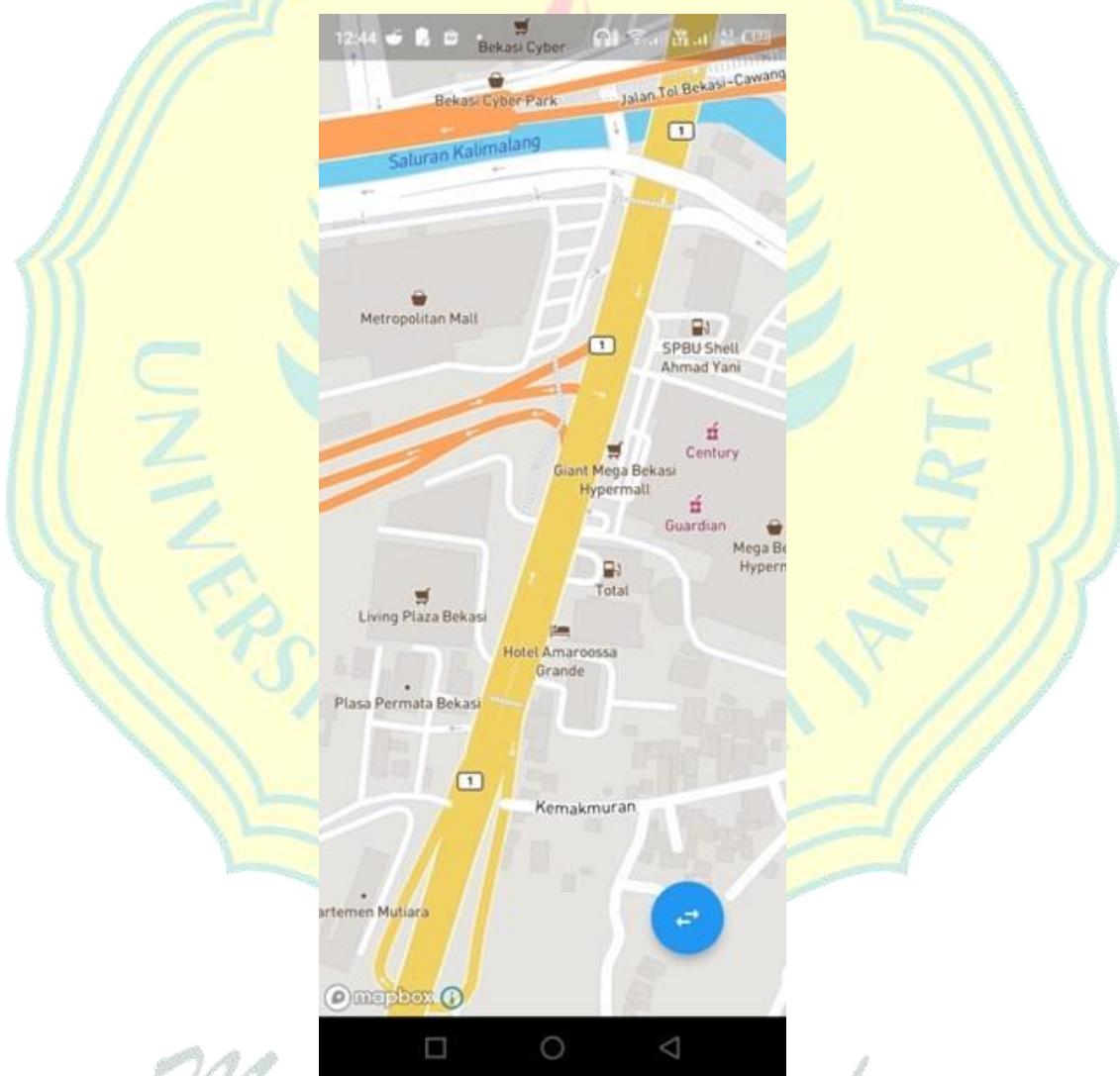
Berbeda dengan OpenStreetMaps, Stasiun Bekasi telah ditandai dengan dua titik POI di peta Mapbox, satu titik POI berada di Jl. Perjuangan dan satu titik berikutnya berada di Jl. Juanda. Sehingga, bagi seseorang yang baru pertama kali menuju ke Stasiun Bekasi pun bisa menandai titik POI Stasiun Bekasi yang terdekat dengan lokasi keberangkatannya. Di dekat POI Stasiun Bekasi juga sudah ada POI untuk toko Kopi Kenangan yang berdiri sejak tahun 2021. Sehingga, temuan ini menyimpulkan setidaknya peta Mapbox telah diperbarui hingga tahun 2021. Berikut merupakan tampilan tangkapan layar peta Mapbox dengan POI Stasiun Bekasi:



Gambar 2.7: Tangkapan Layar Peta Mapbox dengan lokasi Stasiun Bekasi

(b) Pom Bensin Total

Sayangnya, sama seperti OpenStreetMaps, peta Mapbox masih menampilkan titik POI Pom Bensin Total alih-alih Pom Bensin Vivo. Temuan ini bertolak belakang dengan temuan outlet Kopi Kenangan sebelumnya. Sebab, Pom Bensin Total telah tutup secara permanen sejak akhir tahun 2020.



Gambar 2.8: Tangkapan Layar Peta Mapbox yang Menampilkan POI Pom Bensin Total

(c) Kota Cinema Mall Wisma Asri Bekasi

Sama seperti OpenStreetMaps, peta Mapbox sudah menampilkan titik POI Kota Cinema Mall Wisma Asri Bekasi alih-alih Giant Hypermarket.

Sehingga, dapat disimpulkan bahwa peta Mapbox sudah diperbarui setidaknya hingga tahun 2017. Namun, masih belum jelas kenapa peta Mapbox masih menampilkan titik POI Pom Bensin Total alih-alih Pom Bensin Vivo, padahal peta Mapbox sudah menampilkan titik POI untuk Kopi Kenangan yang berdiri sejak tahun 2021. Berikut tampilan layar untuk peta Mapbox yang menampilkan titik POI Kota Cinema Mall Wisma Asri Bekasi:

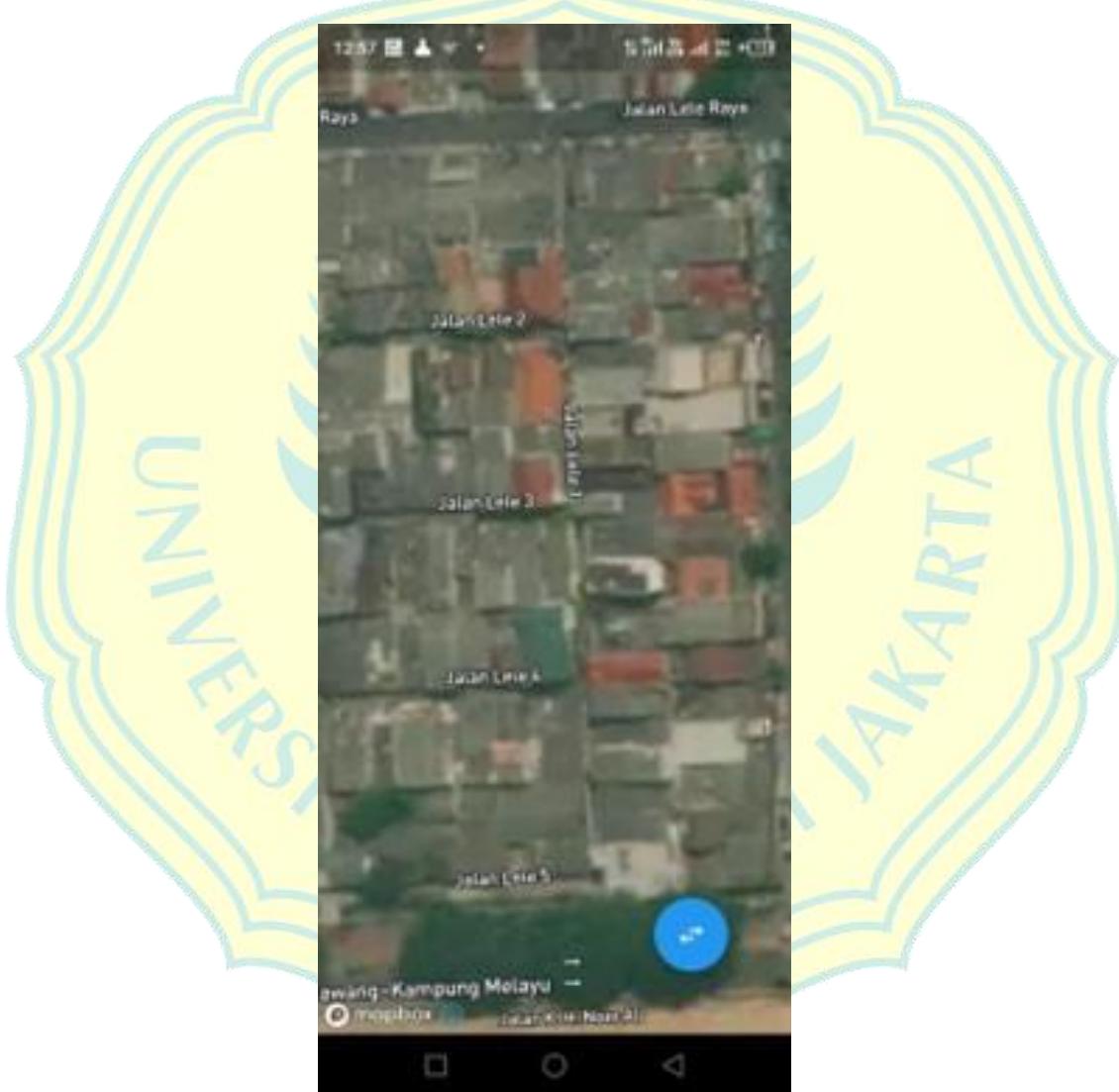


Gambar 2.9: Tangkapan Layar Peta Mapbox yang Menampilkan POI Kota Cinema Mall Wisma Asri Bekasi

(d) Detail Tampilan Pemukiman Warga

Setiap Maps API memiliki keunikannya tersendiri ketika menampilkan

lingkungan pemukiman warga. Sama seperti OpenStreetMaps, lingkungan pemukiman warga hanya ditampilkan nama jalannya saja dan tidak menampilkan nomor rumahnya. Berikut tampilan tangkapan layar untuk pemukiman warga di Jl. Lele, Kayuringin Jaya, Bekasi:



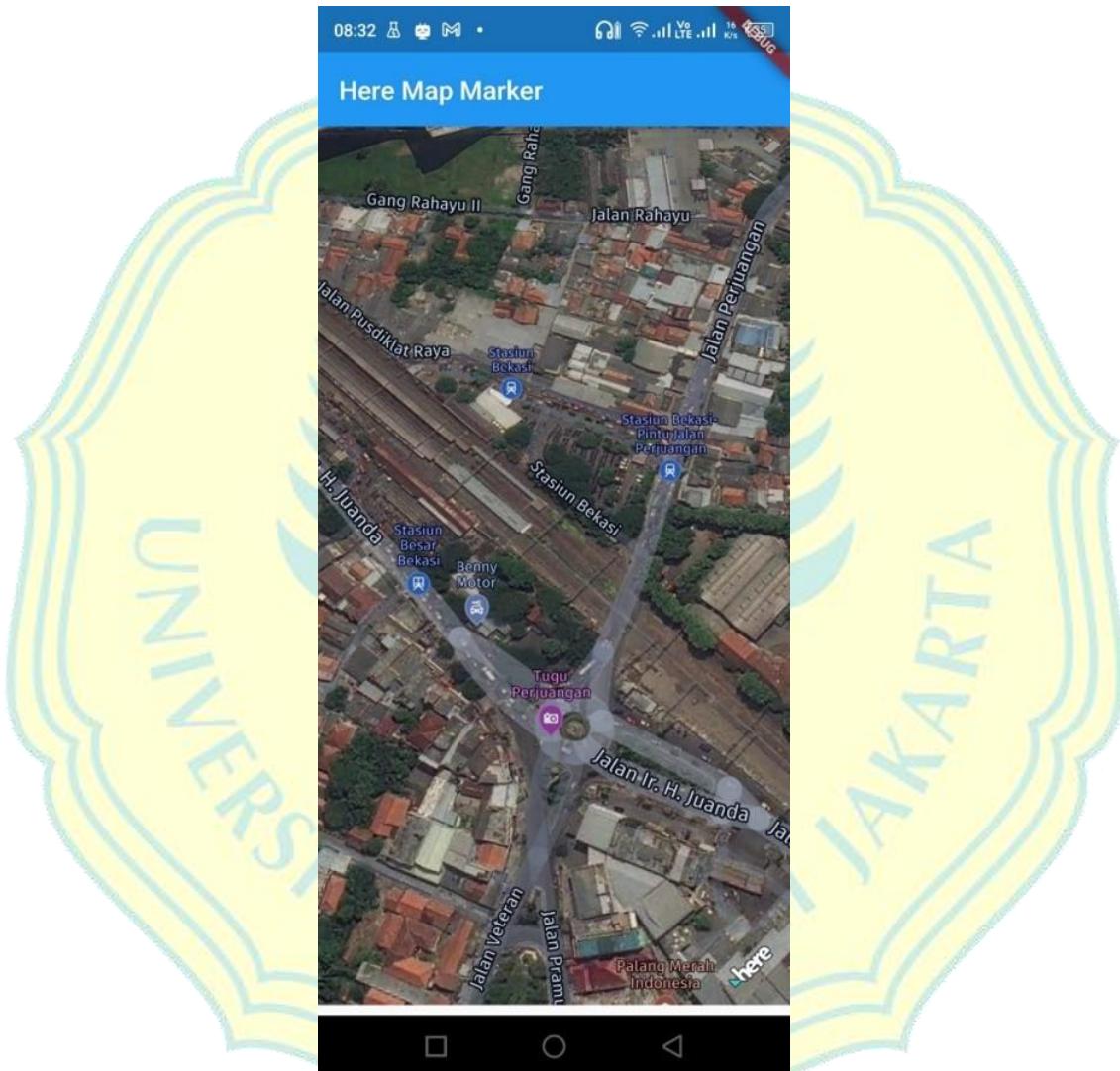
Gambar 2.10: Tangkapan Layar Peta Mapbox yang Menampilkan Pemukiman Warga di Jl. Lele, Kayuringin Jaya, Bekasi

3. HERE Location Service

(a) Stasiun Bekasi

Sama seperti Mapbox, peta Here Location Services juga telah menampilkan dua titik POI untuk masing-masing pintu masuk Stasiun

Bekasi. Berikut tampilan layar peta Here Location Services dengan dua titik POI untuk Stasiun Bekasi:



Gambar 2.11: Tangkapan Layar Peta HERE Location Services dengan lokasi Stasiun Bekasi

(b) Pom Bensin Total

Tidak seperti Mapbox dan OpenStreetMaps, tidak ada titik POI untuk Pom Bensin Total pada peta Here Location Services.



Gambar 2.12: Tangkapan Layar Peta HERE Location Services yang Tidak Menampilkan POI Pom Bensin Total

(c) Kota Cinema Mall Wisma Asri Bekasi

Tidak seperti Mapbox dan OpenStreetMaps, tidak ada titik POI untuk Kota Cinema Mall Wisma Asri Bekasi pada peta Here Location Services.

*Mencerdaskan dan
Memartabatkan Bangsa*



Gambar 2.13: Tangkapan Layar Peta HERE Location Services yang Tidak Menampilkan POI Kota Cinema Mall Wisma Asri Bekasi

(d) Detail Tampilan Pemukiman Warga

Tidak seperti Mapbox dan OpenStreetMaps, peta Here Location Services menampilkan nama jalan dan nomor rumah di lingkungan pemukiman warga.

*Mencerdaskan dan
Memartabatkan Bangsa*



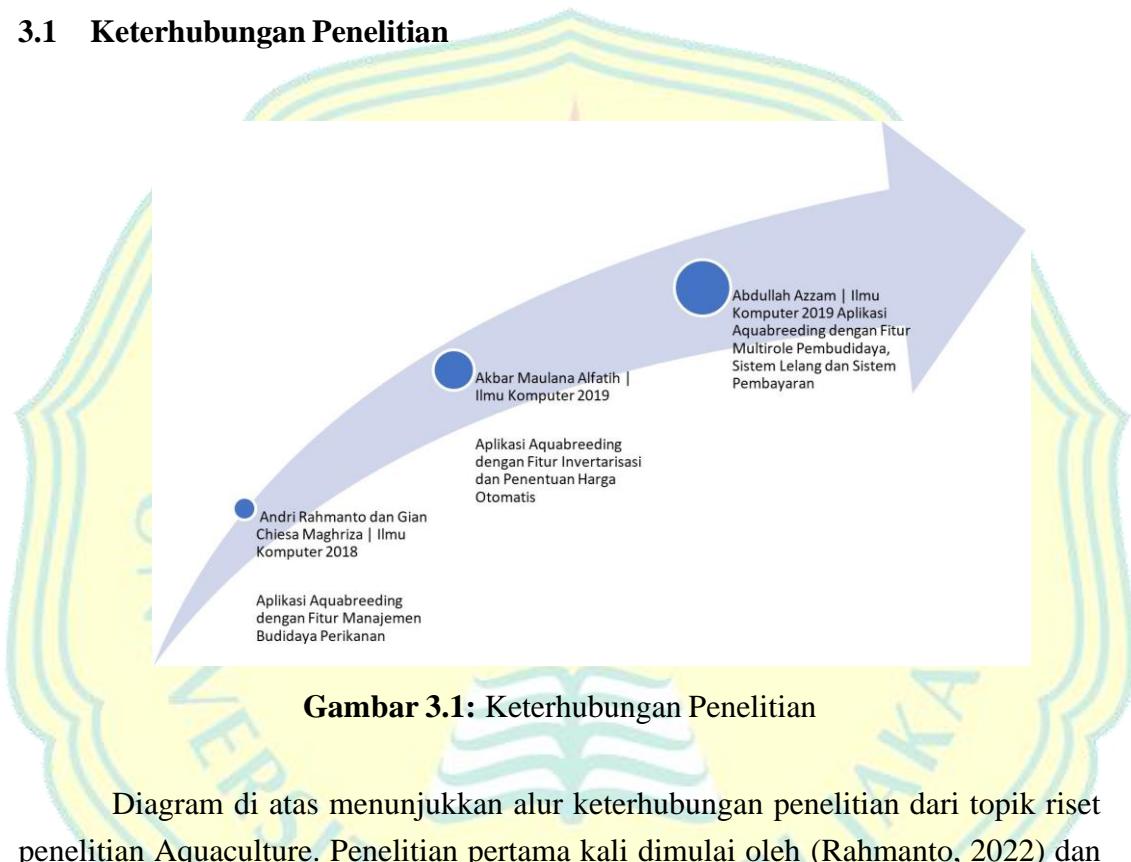
Gambar 2.14: Tangkapan Layar Peta Here Location Services yang Menampilkan Pemukiman Warga di Jl. Lele, Kayuringin Jaya, Bekasi

Mencerdaskan dan
Memartabatkan Bangsa

BAB III

METODOLOGI PENELITIAN

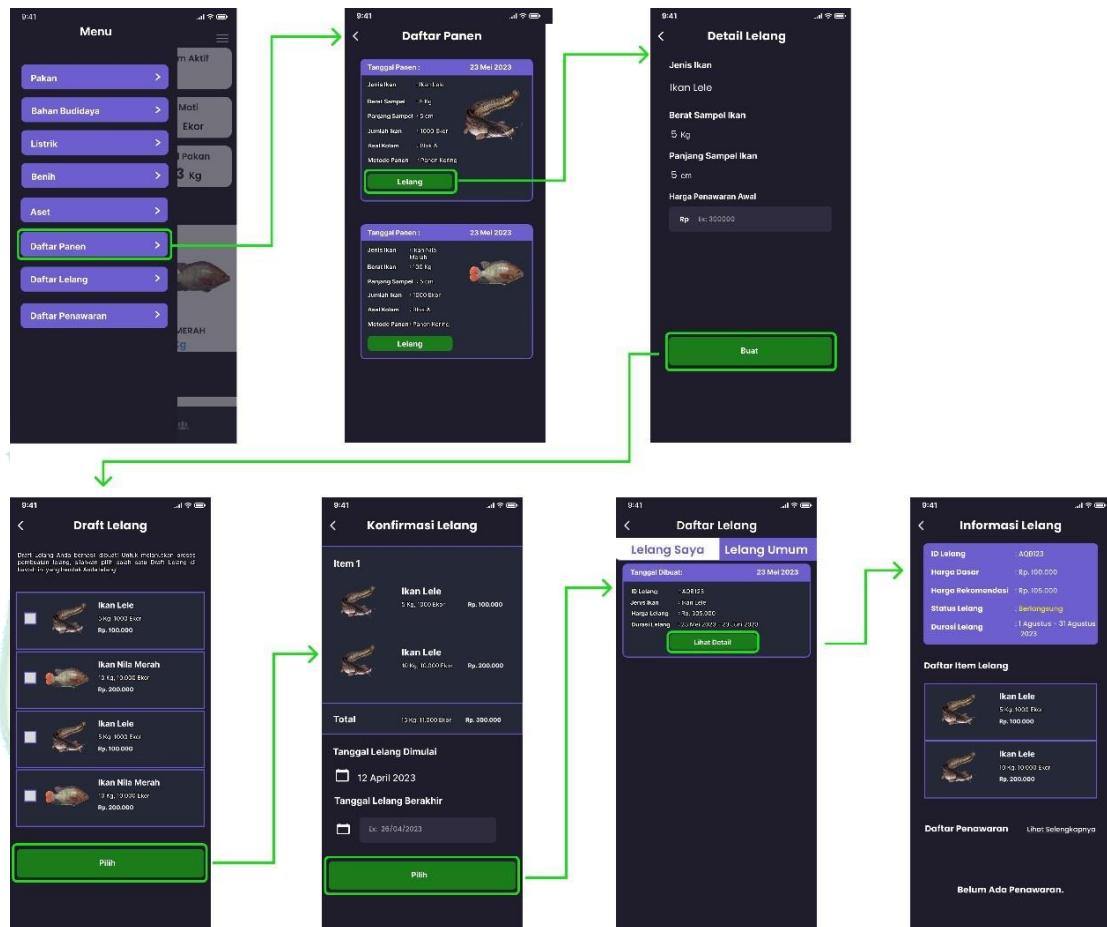
3.1 Keterhubungan Penelitian



Gambar 3.1: Keterhubungan Penelitian

Diagram di atas menunjukkan alur keterhubungan penelitian dari topik riset penelitian Aquaculture. Penelitian pertama kali dimulai oleh (Rahmanto, 2022) dan (Maghriza, 2022) yang membuat aplikasi berbasis Multi-Platform dengan fitur manajemen budidaya perikanan, namun pada penelitian ini belum menerapkan fitur invertarisasi dan penentuan harga otomatis. Kemudian, (Alfatih, 2023) mengembangkan aplikasi yang telah dibuat oleh (Rahmanto, 2022) dan (Maghriza, 2022) dengan menambahkan fitur intervarisasi dan penentuan harga otomatis, namun belum memiliki fitur interkoneksi pembudidaya dan sistem lelang. Fitur interkoneksi pembudidaya diperlukan supaya pembudidaya dapat terhubung satu sama lain, sedangkan sistem lelang diperlukan sebagai transaksi penjualan ikan hasil panen yang menggunakan skema lelang. Pembudidaya dapat membuka lelang untuk jenis ikan yang telah dilelang dengan calon pembelinya sesama pembudidaya.

3.2 Skema Sistem Lelang

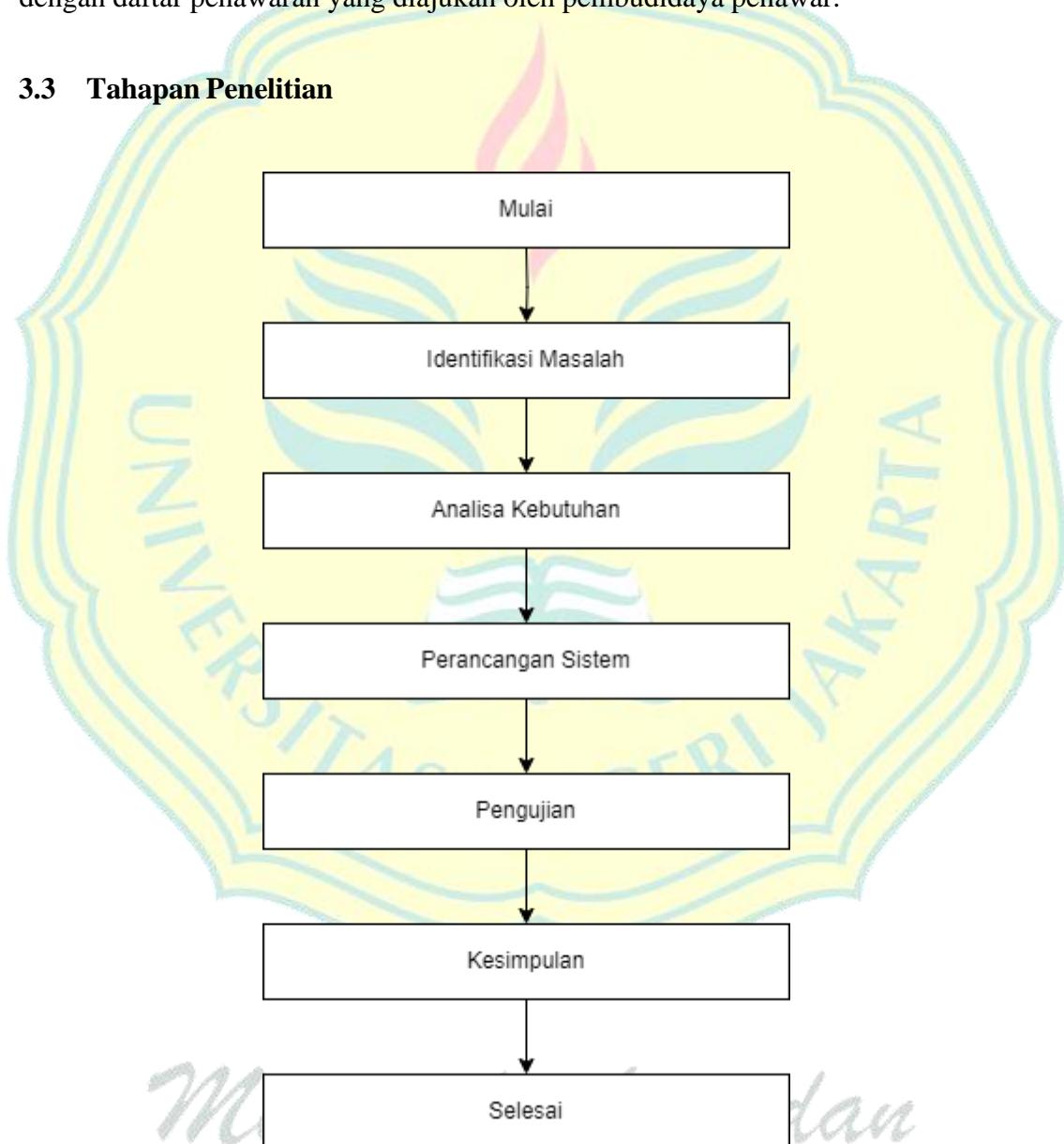


Gambar 3.2: Skema Pembudidaya Membuat Lelang

Skema lelang digunakan untuk menjual ikan hasil panen milik pembudidaya. Skema lelang dipilih agar pembudidaya bisa mendapatkan keuntungan yang cukup besar, adil and masuk akal sebab penawar akan berlomba-lomba mengajukan penawaran harga untuk ikan yang dilelang. Skema lelang dimulai dengan pembudidaya mengisi harga rekomendasi awal untuk ikan yang telah dipanen, kemudian pembudidaya akan diarahkan ke halaman Draft Lelang yang berisi ikan-ikan lain yang sudah dipanen juga, kemudian pembudidaya bisa memilih ikan-ikan yang hendak dilelang secara bersamaan, kemudian pembudidaya mengisi data tanggal lelang berakhir, kemudian pembudidaya membuat lelang tersebut. Ikan yang dilelang bisa berbeda jenis secara bersamaan, oleh karena itu ada halaman Draft Lelang yang berisi daftar berbagai jenis ikan dengan beragam bobot dan

jumlah yang dapat dipilih secara bersamaan, namun apabila pembudidaya hendak melelang satu jenis ikan saja masih dapat dilakukan. Setelah lelang dibuat, pembudidaya dapat melihat daftar lelang yang telah dibuat, kemudian apabila di-klik pembudidaya akan diarahkan ke halaman informasi detail lelang tersebut beserta dengan daftar penawaran yang diajukan oleh pembudidaya penawar.

3.3 Tahapan Penelitian



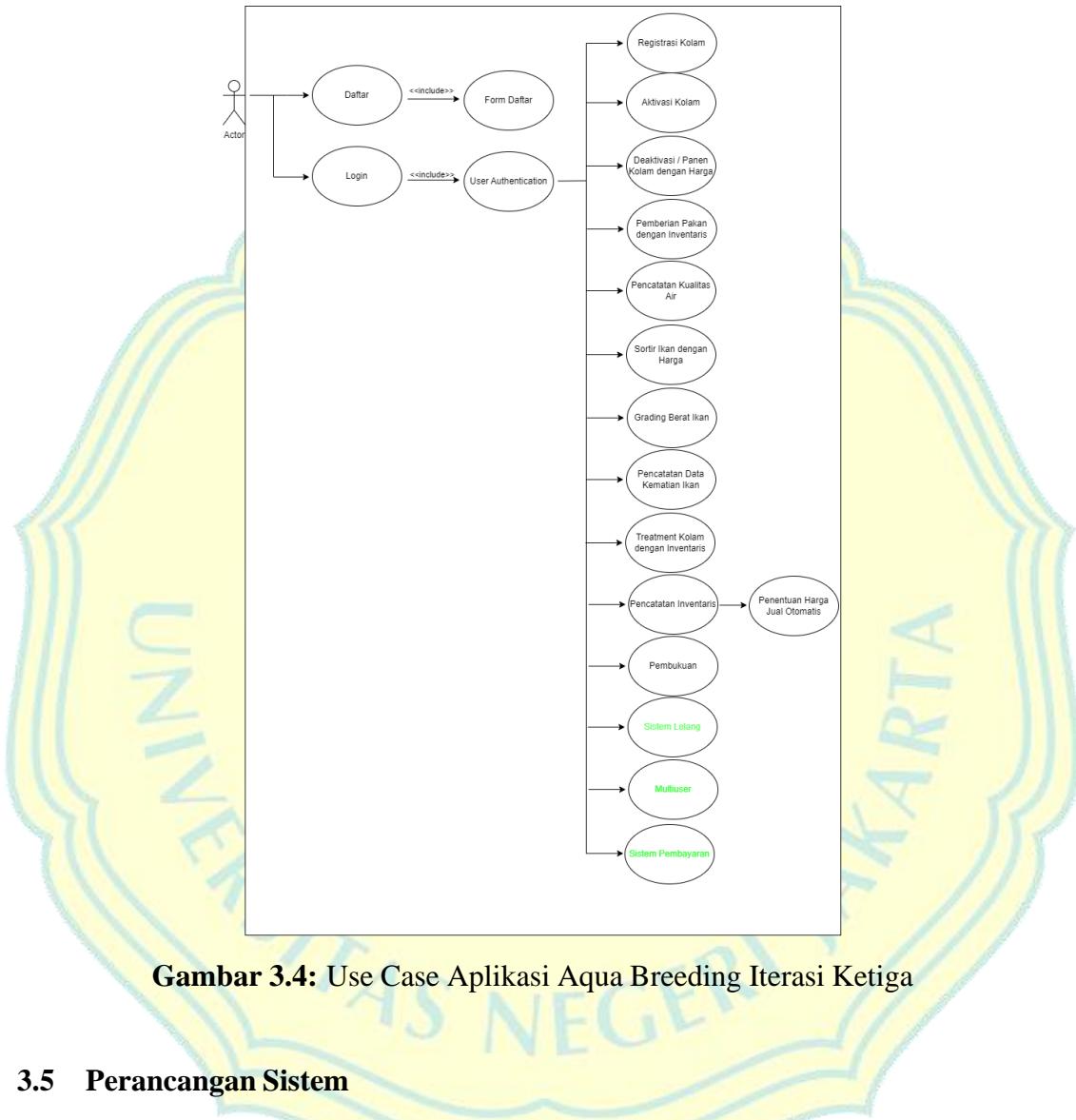
Gambar 3.3: Tahapan Penelitian

Diagram di atas merupakan alur tahapan yang akan diterapkan secara Scrum pada penelitian ini.

3.4 Analisa Kebutuhan

Pada pengembangan aplikasi iterasi ketiga ini, fitur yang akan ditambahkan adalah fitur sistem lelang, sistem pembayaran, serta melakukan perubahan pada fitur multiuser. Fitur sistem lelang merupakan fitur yang membuat pembudidaya dapat melelang ikan-ikan yang telah mereka budidayakan dan sudah memasuki masa panen. Setelah lelang dibuat, maka pembudidaya lain (disebut sebagai pembudidaya penawar) dapat mengajukan penawaran untuk membeli ikan-ikan yang dilelang tersebut. Sistem lelang dipilih sebagai metode transaksi jual-beli sebab para pembudidaya penawar akan menawarkan harga yang bervariasi dan kompetitif sehingga pembudidaya pelelang bisa memilih tawaran yang paling tinggi dan paling masuk akal agar mendapatkan keuntungan sebesar-besarnya dengan cara seadil mungkin. Fitur sistem pembayaran merupakan fitur yang memungkinkan pembudidaya penawar untuk membayarkan sejumlah uang yang telah ia tawarkan apabila tawarannya diterima oleh pembudidaya pelelang. Terakhir, fitur multiuser akan disesuaikan dengan adanya perubahan dan penambahan fitur-fitur baru. Fitur-fitur diatas telah penulis buat menjadi use-case aplikasi agar dapat lebih mudah dipahami. Fitur yang ditulis dengan font berwarna hitam merupakan fitur yang telah ada pada iterasi pertama dan kedua, sedangkan fitur yang ditulis dengan font berwarna hijau merupakan fitur baru yang akan penulis kembangkan pada penelitian ini.

*Mencerdaskan dan
Memartabatkan Bangsa*



3.5 Perancangan Sistem

Metode Scrum dipilih sebagai metode pengembangan aplikasi pada penelitian ini. Komponen-komponen Scrum seperti Product Backlog, Sprint Backlog, Daily Sprint serta Daily Meet akan dibuat dan dilaksanakan seluruhnya agar proses pengembangan aplikasi dapat berjalan lancar, jelas dan tanpa ambiguitas. Berikut penjelasan dan rincian dari masing-masing komponen Scrum:

1. Product Backlog

Garis besar fitur-fitur baru yang akan ditambahkan ke dalam aplikasi disebut sebagai Product Backlog. Garis besar fitur ini kemudian akan dipecah menjadi task-task kecil dan sederhana yang akan dipindahkan pada Sprint Backlog,

yang kemudian akan dikerjakan sesuai dengan skala prioritasnya. Berikut merupakan tabel Product Backlog yang sudah berjalan:

Tabel 3.1: Product Backlog

No	User Story	Priority	Sprint	Status
1	Sistem Lelang	High	1,3,4	Complete
2	Multiuser	Medium	1,3,4	Complete
3	Sistem Pembayaran	Medium	1,3,4	Complete
4	Perbaikan Fitur Aplikasi Versi Kedua	Very High	2	Complete

2. Sprint Backlog

Garis besar fitur yang dipecah menjadi task-task sederhana kemudian dikelompokkan sesuai dengan skala prioritas dan nomor Sprint nya merupakan definisi dari Sprint Backlog. Hal ini berguna agar mempermudah dalam memantau proses pengembangan fitur tersebut. Sprint Backlog bersifat fleksibel sehingga apabila ada task-task baru bisa langsung ditambahkan. Berikut merupakan tabel dari Sprint Backlog yang sudah dan sedang dijalankan.

Tabel 3.2: Sprint Backlog

No	User Story	Task	Status
1	Sistem Lelang	Membuat Desain Mockup fitur Membuat Lelang untuk Pembudidaya Pelelang	Done

No	User Story	Task	Status
		Merancang skema Database untuk fitur Membuat Lelang bagi Pembudidaya Pelelang	Done
		Membuat tabel yang berisikan route server untuk fitur Membuat Lelang bagi Pembudidaya Pelelang	Done
		Mengintegrasikan skema Database Lelang yang telah dibuat dengan database pada iterasi kedua	Done

3. Sprint

Setelah daftar task pada Sprint Backlog telah dibuat dan disepakati bersama, maka proses penggeraan task bisa dimulai sesuai dengan urutan Sprint dan skala prioritasnya.

- (a) Desain Mockup UI/UX Sistem Lelang - Membuat Lelang untuk Pembudidaya
- (b) Berikut merupakan hasil rancangan skema Database Lelang sekaligus Class Diagram untuk fitur Membuat Lelang bagi Pembudidaya Pelelang:

The diagram illustrates the database schema for the 'Pembudidaya Membuat Lelang' feature. It consists of three tables:

- Bid List**: Contains columns for id (PK, ObjectId), user_log_id (FK, ObjectId), auction_id (FK, ObjectId), bid_price (Int NOT NULL), date_to_claim (Date NOT NULL), created_at (Date NOT NULL), and updated_at (Date NOT NULL).
- Auction List**: Contains columns for id (PK, ObjectId), draft_auction_list_id (FK, ObjectId), auction_item_list (FK, ObjectId), price (Int NOT NULL), recommendation_price (Int NOT NULL), highest_bid (Int NOT NULL), lowest_bid (Int NOT NULL), auction_start_date (Date NOT NULL), auction_end_date (Date NOT NULL), created_at (Date NOT NULL), and updated_at (Date NOT NULL).
- Draft Auction List**: Contains columns for id (PK, ObjectId), origin_pond_id (FK, ObjectId), fish_log_id (FK, ObjectId), fish_transfer_id (FK, ObjectId), fish_type (String NOT NULL), fish_amount (Int NOT NULL), sample_weight (Int NOT NULL), sample_long (Int NOT NULL), price (Int NOT NULL), created_at (Date NOT NULL), and updated_at (Date NOT NULL).

Gambar 3.5: Skema Database untuk Fitur Pembudidaya Membuat Lelang

- (c) Berikut merupakan tabel yang berisi route server untuk fitur Membuat Lelang bagi Pembudidaya Pelelang:

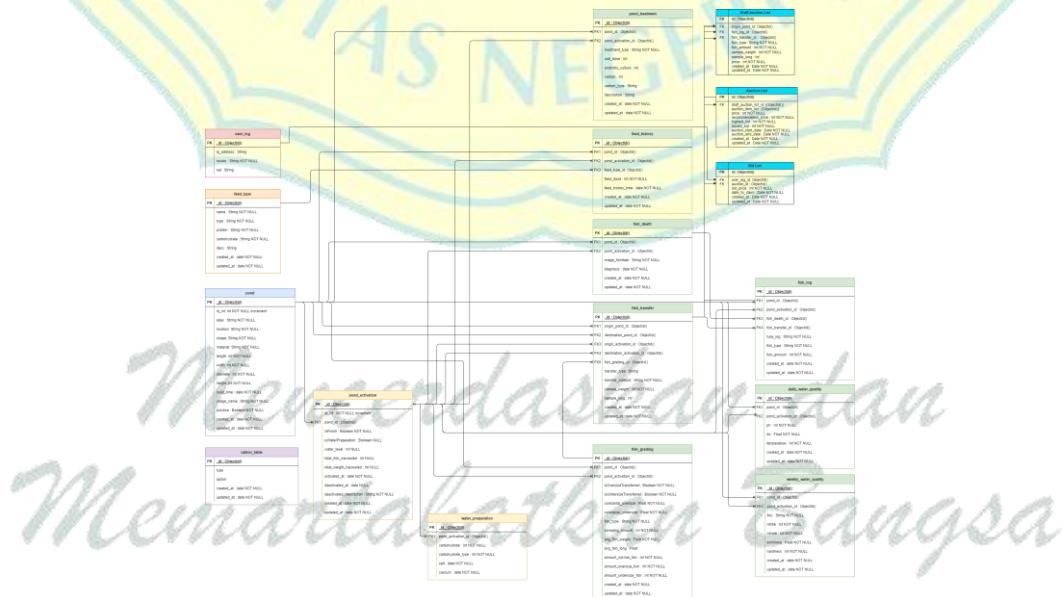
Tabel 3.4: Tabel Route Server yang berisi Route Server untuk Fitur Membuat Lelang

Group	Endpoint	HTTP Status	Purpose
Sistem Lelang – Membuat Lelang	/auction/draft/create	POST	Membuat draft lelang
	/auction/create	POST	Membuat lelang
	/auction/list/userId	GET	Mengambil daftar lelang untuk role yang berbeda sesuai dengan value userId yang diberikan pada parameter.

*Mencerdaskan dan
Memartabatkan Bangsa*

Group	Endpoint	HTTP Status	Purpose
	/auction/details/ <i>auction_id</i>	GET	Mengambil informasi lebih detail mengenai sebuah lelang sesuai dengan value <i>auction_id</i> yang diberikan pada parameter.
	/auction/bid- list/ <i>auction_id</i>	GET	Mengambil daftar penawaran yang diberikan oleh Pembudidaya Penawar pada sebuah lelang.

(d) Integrasi skema Database Lelang dengan Skema Database Iterasi 2:

**Gambar 3.6:** Integrasi Skema Database Lelang dengan Skema Database Iterasi 2

4. Sprint Review

Sprint Review akan dilaksanakan setiap minggu setelah proses penggerjaan Sprint dimulai. Sprint Review bertujuan untuk melaporkan proses penggerjaan Sprint, melaporkan kendala yang dialami ketika penggerjaan Sprint, dan pemberian masukan berupa revisi maupun task baru.

5. Deploy Sistem

Setelah seluruh daftar task pada Sprint Backlog telah selesai dikerjakan, maka langkah selanjutnya yakni melakukan proses deploy agar aplikasi dapat diujikan menggunakan Unit Testing.

3.6 Pengujian

Pengujian akan dilakukan melalui empat tahap Unit Testing. Pada tahap pertama, fokus pengujian akan difokuskan pada fitur-fitur yang telah diperbaiki pada aplikasi versi kedua selama Sprint 2. Tahap kedua hingga tahap keempat akan menitikberatkan pada pengujian fitur-fitur baru yang ditambahkan dalam aplikasi versi ketiga, termasuk fitur lelang, sistem keuangan, dan kemampuan multiuser bagi pembudidaya. Berikut merupakan skenario pengujian unit testing dari aplikasi versi ketiga.

Jenis Fitur	Skenario Pengujian
Perhitungan FCR, Survival Rate dan Fungsi Mengirim Foto pada fitur Panen	Pembudidaya mengisi data panen seperti jumlah ikan yang dipanen, total berat ikan yang dipanen, memilih atau mengambil foto ikan yang dipanen, kemudian pembudidaya klik tombol Panen
	Ketika pembudidaya klik detail musim budidaya, akan ditampilkan jumlah FCR, Survival Rate dan menampilkan foto ikan hasil panen.
	Jika foto nilai FCR dan Survival Rate bernilai 0 dan foto ikan hasil panen tidak tampil padahal data panen sudah diisi secara lengkap, berarti fungsi untuk menghitung FCR dan Survival Rate serta fungsi untuk mengirim foto di <i>backend</i> masih salah.

Jenis Fitur	Skenario Pengujian
Pencatatan Kualitas Air Harian	Pembudidaya klik detail kolam, kemudian pembudidaya klik tab kondisi air.
	Pembudidaya klik musim budidaya yang sedang berlangsung. Kemudian, pembudidaya klik <i>floating action button</i> plus untuk mengisikan entri kualitas air harian baru.
	Pemmbudidaya mengisikan data kualitas air seperti pH, DO dan Suhu Air. Apabila sudah, pembudidaya akan diarahkan kembali ke halaman kualitas air harian. Bila muncul <i>widget card</i> yang berisikan data kualitas air yang memiliki nilai yang sama dengan nilai yang baru saja diinput, maka fitur pencatatan kualitas air harian telah berfungsi.
Pencatatan Kematian Ikan	Pembudidaya klik detail kolam, kemudian pembudidaya klik detail musim budidaya yang sedang berlangsung.
	Pembudidaya scroll tampilan layar hingga menemukan tombol Rekapitulasi Kematian, di mana pembudidaya akan diarahkan ke halaman data rekapitulasi kematian ikan. Kemudian pembudidaya klik tombol <i>Entry</i> Kematian Ikan.
	Pembudidaya memilih jenis ikan yang mati, kemudian mengisikan jumlah kematian ikan dan diagnosa kematian ikan, kemudian klik Submit. Bila pada halaman rekapitulasi kematian ikan bertambah <i>widget card</i> yang berisikan data kematian ikan yang baru saja diinput oleh pembudidaya, maka fungsi pencatatan kematian ikan telah berfungsi.
Memperbaiki fungsi Aktivasi Kolam untuk dapat menerima data <i>double</i> pada tinggi air	Pembudidaya klik detail kolam, kemudian pembudidaya klik musim budidaya yang sedang berlangsung.

Jenis Fitur	Skenario Pengujian
	Pembudidaya akan diarahkan ke halaman entri data aktivasi kolam. Pembudidaya mengisikan data seperti bentuk kolam, dimensi kolam, mengisikan data <i>double</i> pada kolom tinggi air, memilih jenis ikan yang akan dimulai musim budidaya nya, dan klik tombol Aktivasi.
	Pembudidaya akan diarahkan kembali ke halaman detail kolam. Apabila kolam yang baru saja diinput datanya berhasil diaktifasi, maka fungsi aktivasi kolam yang telah diperbaiki sudah berfungsi.
Membuat Lelang	Pembudidaya klik halaman Rekap Panen, kemudian pembudidaya klik salah satu <i>widget card</i> yang hendak dilelang
	Pembudidaya akan diarahkan ke halaman Konfirmasi Detail Lelang. Pembudidaya mengisikan nominal harga penawaran awal yang diinginkan, kemudian klik Konfirmasi.
	Di halaman Lelang Grosir, pembudidaya klik <i>checkbox</i> dari data Lelang Grosir yang hendak dilelang, bisa hanya 1 atau lebih untuk membuat paket lelang.
	Pembudidaya akan diarahkan ke halaman Konfirmasi Detail Lelang, mengisikan data seperti nama lelang, tanggal lelang dimulai dan tanggal lelang berakhir, kemudian klik tombol Konfirmasi.
	Pembudidaya menuju halaman Lelang Saya, bila tercipta satu <i>widget card</i> baru berisi data lelang yang baru saja diinput oleh pembudidaya, maka fitur membuat lelang telah berfungsi.
Mengajukan Penawaran	Pembudidaya penawar <i>switch</i> ke halaman Daftar Lelang, kemudian klik salah satu <i>widget card</i> lelang yang diminati.

Jenis Fitur	Skenario Pengujian
	Pembudidaya akan diarahkan ke halaman Detail Lelang, kemudian mengisikan nominal jumlah penawaran yang diinginkan. Bila muncul pesan berhasil, maka penawaran berhasil diajukan.
Menerima Penawaran	Pembudidaya klik halaman Lelang Saya, kemudian pembudidaya klik salah satu <i>widget card</i> lelang yang hendak dilihat penawarannya.
	Pembudidaya akan diarahkan ke halaman Detail Lelang. Pembudidaya dapat melihat daftar penawaran di paling bawah halaman.
	Pembudidaya klik tombol Terima Penawaran di salah satu <i>wdiget card</i> dengan penawaran paling menarik.
	Akan tampil pop-up untuk mengonfirmasi penerimaan penawaran, bila pembudidaya yakin silahkan klik Ya.
	Apabila tampilan Detail Lelang di bagian bawah berubah dari Daftar Penawaran menjadi Penawar Terpilih, maka fitur telah berfungsi dengan baik.
Topup	Pembudidaya klik halaman Menu Page, scroll hingga menemukan tombol Topup, lalu klik tombol tersebut.
	Pembudidaya akan diarahkan ke halaman Topup. Pembudidaya mengisikan nominal topup yang diinginkan, kemudian masukkan foto bukti transfer.
	Klik tombol Topup, bila muncul pesan berhasil maka permintaan topup berhasil diajukan.

*Mencerdaskan dan
Memartabatkan Bangsa*

BAB IV

HASIL DAN PEMBAHASAN

4.1 Perancangan Sistem Dengan Scrum

Pengembangan aplikasi Aqua Breeding direncanakan dengan menerapkan metode Scrum. Dalam pendekatan ini, proses pengembangan dilakukan secara bertahap yang disebut sebagai Sprint. Terdapat 4 Sprint dalam penelitian ini, yang dikerjakan secara bertahap berdasarkan skala prioritas dan tingkat kesulitannya. Sebelum memulai setiap Sprint, dilakukan perencanaan Sprint Backlog yang diperoleh dari Product Backlog yang telah direncanakan dan disetujui sebelumnya. Rincian laporan untuk setiap Sprint dalam proses pengembangan sistem dapat ditemukan di bawah ini:

4.1.1 Sprint 1

Sprint 1 dilaksanakan pada tanggal 14 Juli 2023 - 7 Agustus 2023. Detail dari Sprint 1 ini adalah mengerjakan tugas yang ada pada Sprint 1 Backlog di tabel berikut.

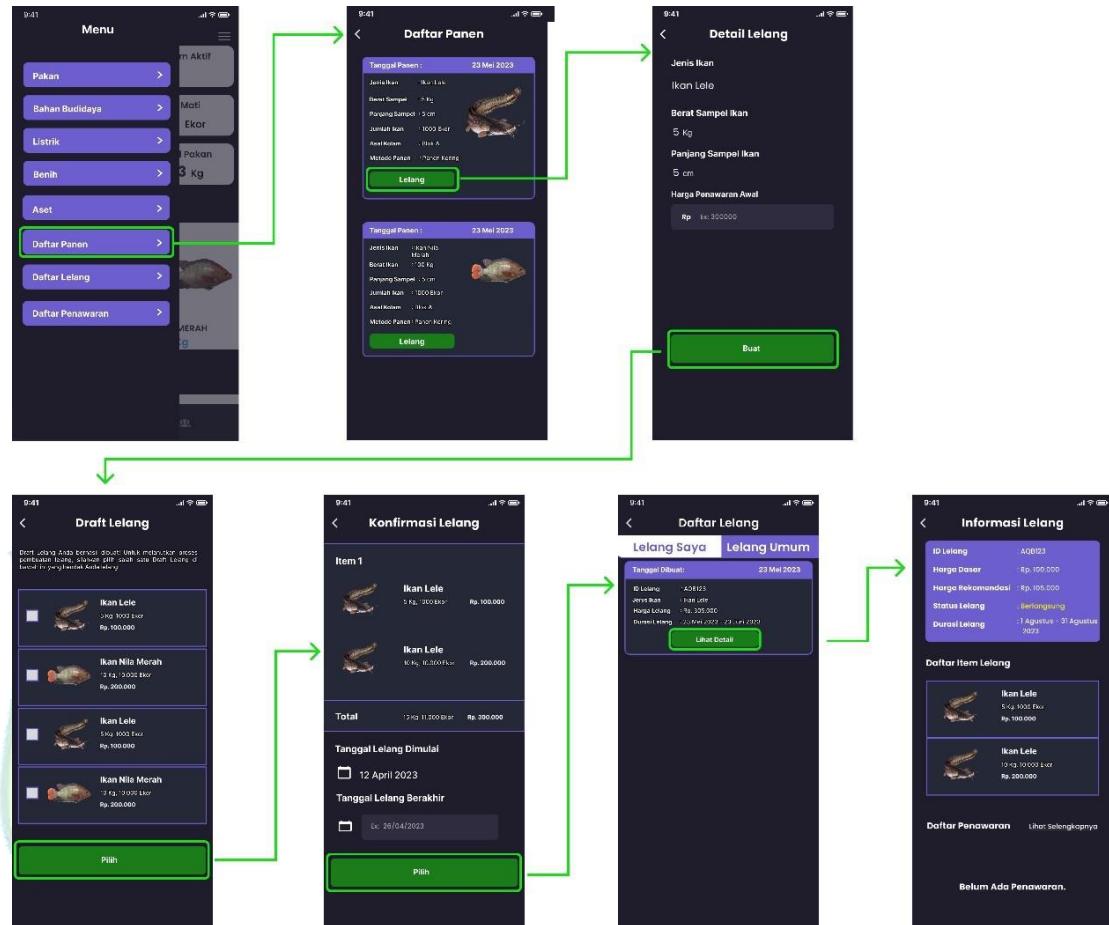
Tabel 4.1: Sprint 1 Backlog

No	User Story	Task	Status
1	Sistem Lelang	Membuat Desain Mockup fitur Membuat Lelang untuk Pembudidaya Pelelang	Done

No	User Story	Task	Status
		Merancang skema Database untuk fitur Membuat Lelang bagi Pembudidaya Pelelang	Done
		Membuat tabel yang berisikan route server untuk fitur Membuat Lelang bagi Pembudidaya Pelelang	Done
		Mengintegrasikan skema Database Lelang yang telah dibuat dengan database pada iterasi kedua	Done

*Mencerdaskan dan
Memajukan Bangsa*

Berikut merupakan desain *mockup* sekaligus flow aplikasi untuk fitur Membuat Lelang.



Gambar 4.1: Skema Pembudidaya Membuat Lelang

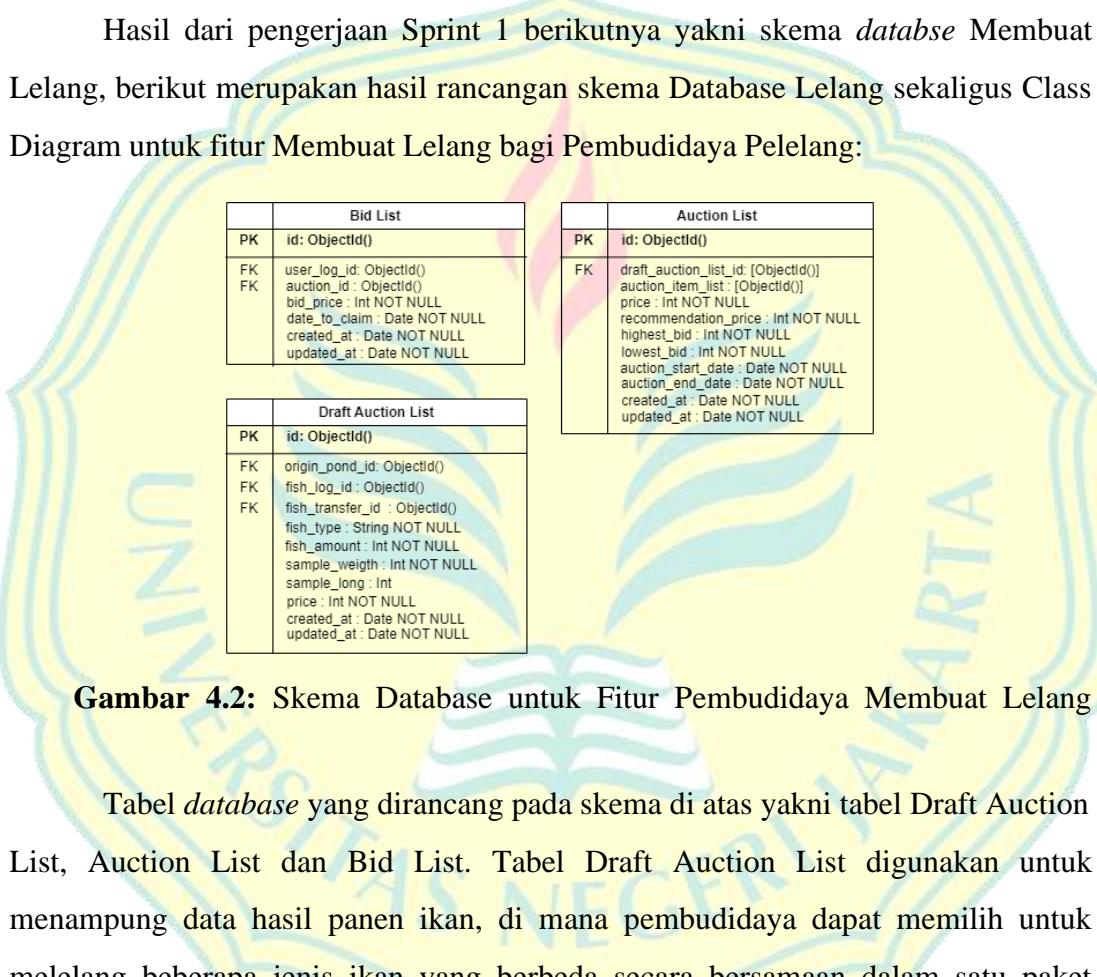
Di halaman dashboard, terdapat menu *hamburger* yang apabila ditekan akan menampilkan daftar menu inventaris, daftar panen, daftar lelang dan daftar penawaran. Apabila pembudidaya hendak melelang hasil panen nya, maka *flow* nya yakni pembudidaya harus menekan tombol daftar panen terlebih dahulu, kemudian di halaman daftar panen akan menampilkan daftar *widget card* yang berisi informasi ikan yang telah dipanen seperti informasi jenis ikan, berat sampel, panjang sampel, jumlah ikan, asal kolam, metode panen serta tombol berwarna hijau bertuliskan "Lelang". Apabila tombol Lelang ditekan, maka pembudidaya akan diarahkan ke halaman detail lelang yang berisi informasi jenis ikan, berat sampel ikan dan panjang sampel ikan yang hendak dilelang, serta satu *widget textfield* untuk memasukkan harga penawaran awal lelang yang diinginkan oleh pembudidaya.

Ketika pembudidaya menekan tombol Buat, maka data ikan yang hendak dilelang tersebut akan dikirimkan ke *database* untuk disimpan di tabel Draft Auction List, dan apabila proses pengiriman data ke *database* tidak ada *error*, maka pembudidaya akan diarahkan ke halaman Draft Lelang. Di halaman Draft Lelang, akan menampilkan daftar data-data ikan yang hendak dilelang yang disimpan di tabel Draft Auction List. Pada masing-masing data ikan, ada kolom *checkbox* untuk memilih lebih dari satu ikan hasil panen yang hendak dilelang untuk kemudian dibuatkan paket lelang. Pembudidaya tetap dapat hanya melelang satu jenis ikan saja cukup dengan menekan satu kali pada satu kolom *checkbox* saja. Kemudian, ketika pembudidaya menekan tombol Pilih, maka pembudidaya akan diarahkan ke halaman Konfirmasi Lelang, di mana pada halaman tersebut akan menampilkan data ikan terpilih untuk dipanen seperti jenis ikan, berat ikan, jumlah ikan, harga penawaran awal lelang yang diinginkan pembudidaya, serta pembudidaya akan diminta untuk memasukkan tanggal lelang dimulai dan tanggal lelang berakhir. Apabila pembudidaya hendak melelang lebih dari satu jenis ikan, maka akan dihitung pula total berat, total jumlah ikan dan total harga penawaran awal lelang. Tanggal lelang dimulai akan otomatis diatur pada tanggal saat halaman Konfirmasi Lelang diakses. Kemudian, ketika pembudidaya menekan tombol Pilih, maka data ikan yang hendak dipanen akan dikirimkan ke *database* untuk disimpan di tabel Auction List, apabila proses pengiriman data ke *database* berjalan dengan lancar tanpa *error* maka pembudidaya akan diarahkan ke halaman Daftar Lelang.

Halaman Daftar Lelang dapat diakses melalui Daftar Menu yang akan muncul ketika menekan menu *hamburger*. Di halaman daftar lelang, akan menampilkan daftar *widget card* yang menampilkan informasi seperti ID lelang, jenis ikan yang dilelang, harga penawaran awal lelang, tanggal lelang dimulai dan tanggal lelang berakhir. Apabila terdapat lebih dari satu jenis ikan yang dilelang, maka akan disebutkan semua jenis ikan pada *widget card* tersebut. Ketika pembudidaya menekan tombol Lihat Detail, maka pembudidaya akan diarahkan ke halaman Informasi Lelang, yang akan menampilkan informasi detail dari lelang

yang telah dibuat, seperti informasi ID Lelang, harga rekomendasi lelang dari fitur rekomendasi harga, harga penawaran awal, status lelang, durasi lelang, daftar item lelang, serta daftar penawaran.

Hasil dari penggerjaan Sprint 1 berikutnya yakni skema *database* Membuat Lelang, berikut merupakan hasil rancangan skema Database Lelang sekaligus Class Diagram untuk fitur Membuat Lelang bagi Pembudidaya Pelelang:



Bid List	
PK	id: ObjectId()
FK	user_log_id: ObjectId()
FK	auction_id: ObjectId()
	bid_price : Int NOT NULL
	date_to_claim : Date NOT NULL
	created_at : Date NOT NULL
	updated_at : Date NOT NULL

Auction List	
PK	id: ObjectId()
FK	draft_auction_list_id: ObjectId()
	auction_item_list: [ObjectId()]
	price : Int NOT NULL
	recommendation_price : Int NOT NULL
	highest_bid : Int NOT NULL
	lowest_bid : Int NOT NULL
	auction_start_date : Date NOT NULL
	auction_end_date : Date NOT NULL
	created_at : Date NOT NULL
	updated_at : Date NOT NULL

Draft Auction List	
PK	id: ObjectId()
FK	origin_pond_id: ObjectId()
FK	fish_log_id: ObjectId()
FK	fish_transfer_id: ObjectId()
	fish_type : String NOT NULL
	fish_amount : Int NOT NULL
	sample_weight : Int NOT NULL
	sample_long : Int
	price : Int NOT NULL
	created_at : Date NOT NULL
	updated_at : Date NOT NULL

Gambar 4.2: Skema Database untuk Fitur Pembudidaya Membuat Lelang

Tabel *database* yang dirancang pada skema di atas yakni tabel Draft Auction List, Auction List dan Bid List. Tabel Draft Auction List digunakan untuk menampung data hasil panen ikan, di mana pembudidaya dapat memilih untuk melelang beberapa jenis ikan yang berbeda secara bersamaan dalam satu paket lelang. Pembudidaya tetap dapat melelang satu hasil panen ikan saja, namun dengan melelang beberapa jenis hasil panen ikan secara bersamaan dapat meningkatkan keuntungan bagi pembudidaya, memudahkan pembudidaya dalam membungkus ikan untuk dikirim ke penawar terpilih, terutama apabila penawar terpilih merupakan restoran yang menyajikan menu ikan. Pada tabel Draft Auction List, terdapat *field* origin_pond_id untuk menginformasikan ikan tersebut hasil panen dari kolam yang mana, kemudian fish_log_id dan fish_transfer_id untuk referensi id jenis ikan yang telah dipanen dan hendak dilelang, fish_type untuk mencatat jenis ikan, fish_amount untuk mencatat jumlah ikan yang dipanen dan hendak dilelang, sample_weight dan

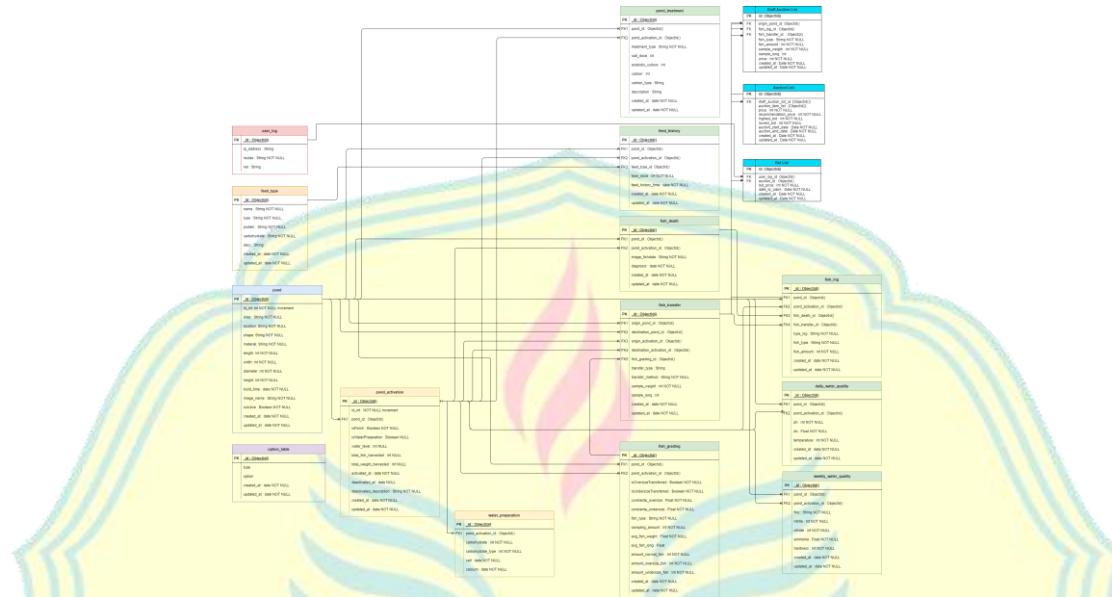
sample_long untuk mencatat berat dan ukuran dari sampel ikan yang dipanen dan hendak dilelang, price untuk mencatat harga lelang yang ditentukan oleh pembudidaya, serta created_at dan updated_at untuk mencatat tanggal pembuatan dan pembaruan ketika data tersebut dimasukkan ke *database*.

Tabel Auction List dirancang untuk menampung data ikan-ikan yang telah dipilih untuk dilelang, baik satuan maupun dalam bentuk paket lelang beberapa jenis ikan sekaligus. *Field* yang ada pada tabel Auction List yakni draft_auction_list_id yang merupakan referensi ke daftar objectId dari data ikan hasil panen yang sebelumnya dimasukkan ke dalam tabel Draft Auction List, kemudian *field* auction_item_list yang menyimpan daftar ikan hasil panen yang dilelang (tetapi akan menyimpan dalam bentuk daftar walau ikan yang dipanen hanya satu jenis saja), lalu *price* yang menyimpan data harga lelang ikan, recommendation_price yang menyimpan data harga lelang ikan yang direkomendasikan oleh fitur rekomendasi harga jual, lalu *field* highest_bid dan lowest_bid untuk menyimpan data penawaran tertinggi dan penawaran terendah, terakhir *field* auction_start_date dan auction_end_date yang mencatat tanggal lelang dimulai dan tanggal lelang berakhir.

Tabel terakhir yakni tabel Bid List, yang akan menyimpan penawaran-penawaran yang masuk dari pembudidaya ikan yang lain. Tabel ini memiliki *field* user_log_id untuk mencatat identitas pembudidaya yang mengajukan penawaran, kemudian auction_id untuk referensi ke objectId dari data ikan yang dilelang, bid_price untuk mencatat harga yang ditawarkan, date_to_claim yang mencatat tanggal pengambilan ikan, serta created_at dan updated_at untuk mencatat tanggal pembuatan dan pembaruan data penawaran yang dimasukkan ke *database*.

Setelah skema *database* lelang telah dibuat, maka tabel-tabel tersebut perlu diintegrasikan dengan skema *database* sebelumnya agar skema *database* nya lengkap. Berikut merupakan hasil integrasi tabel-tabel Lelang dengan skema *database*:

Memerdayakan dan Memartabatkan Bangsa



Gambar 4.3: Integrasi Skema Database Lelang dengan Skema Database Iterasi 2

Hasil dari penggerjaan Sprint 1 yang terakhir yakni tabel yang berisi route server untuk fitur Membuat Lelang bagi Pembudidaya Pelelang:

Tabel 4.3: Tabel Route Server yang Berisi Route untuk Fitur Membuat Lelang

Group	Endpoint	HTTP Status	Purpose
Sistem Lelang – Membuat Lelang	/auction/draft/create	POST	Membuat draft lelang
	/auction/create	POST	Membuat lelang
	/auction/list/userId	GET	Mengambil daftar lelang untuk peran yang berbeda berdasarkan nilai userId pada parameter

Group	Endpoint	HTTP Status	Purpose
	/auction/details/ <i>auction_id</i>	iGET	Mengambil informasi detail tentang suatu lelang berdasarkan nilai <i>auction_id</i> pada parameter
	/auction/bid-list/ <i>auction_id</i>	GET	Mengambil daftar penawaran dari Pembudidaya Penawar pada suatu lelang

4.1.2 Sprint 2

Setelah menyelesaikan serangkaian *task* dalam Sprint 1 *Backlog*, peneliti melakukan evaluasi terhadap fungsionalitas aplikasi Aqua Breeding versi kedua. Dalam proses ini, peneliti menemukan beberapa fitur yang masih memerlukan perbaikan agar aplikasi dapat berfungsi sesuai dengan harapan, terutama jika dibandingkan dengan aplikasi versi pertama. Berdasarkan temuan ini, peneliti membuat daftar fitur yang perlu diperbaiki sebagai dasar untuk melaksanakan Sprint 2.

Sprint 2 dilaksanakan mulai dari tanggal 15 September 2023 hingga 14 Oktober 2023. Fokus utama Sprint 2 adalah menyelesaikan *task* yang berkaitan dengan perbaikan fitur-fitur yang teridentifikasi sebelumnya. Tabel berikut merinci

detail dari Sprint 2 Backlog, yang mencakup daftar *task* yang harus diselesaikan.

Tentu saja, pelaksanaan Sprint 2 ini bertujuan untuk meningkatkan kualitas dan kinerja aplikasi Aqua Breeding versi ketiga. Dengan merinci *task* yang perlu diperbaiki, diharapkan setiap aspek yang mengalami kendala pada versi sebelumnya dapat diatasi dengan baik. Sprint 2 menjadi bagian integral dari upaya pengembangan aplikasi ini, yang secara keseluruhan ditujukan untuk memberikan solusi yang lebih baik dan efisien bagi para pengguna, terutama pembudidaya ikan.

Tabel 4.5: Sprint 2 Backlog

No	User Story	Task	Status
1	Panen	Memperbaiki perhitungan FCR Memperbaiki perhitungan <i>survival rate</i> Menambahkan fungsi untuk mengirimkan foto	Done Done Done
2	Pencatatan Kualitas Air Harian	Memperbaiki fitur pencatatan kualitas air harian	Done
3	Pencatatan Kematian	Memperbaiki fitur pencatatan kematian ikan	Done
4	Aktivasi Kolam	Memperbaiki fitur aktivasi kolam agar dapat mengirimkan data tinggi air dalam format <i>double</i>	Done

Hasil dari penggerjaan Sprint 2 yakni:

1. Memperbaiki Perhitungan FCR

Fungsi getFCR (lihat Lampiran II poin 2.1) dipanggil pada API Panen, yang akan menghitung FCR (Feed Conversion Rate) dari satu musim budidaya untuk mengetahui apakah pemberian pakan pada ikan sudah optimal atau belum. Fungsi ini akan mengambil riwayat pakan dan menghitung pertumbuhan ikan, kemudian membagi total penggunaan pakan dengan total bobot ikan untuk menghitung FCR (*Feed Conversion Ratio*) nya. Semakin kecil nilai FCR nya, artinya semakin optimal penggunaan pakannya, sebab penggunaan pakan yang sedikit namun mampu membesarkan ukuran ikan hingga layak dipasarkan.

2. Memperbaiki Perhitungan Survival Rate

Fungsi GET pada Lampiran II poin 2.2 merupakan *query pipeline* yang digunakan untuk mengambil data dari MongoDB secara spesifik. *Query pipeline* mirip dengan *query* pada SQL, yakni mengembalikan data sesuai dengan yang dideskripsikan pada *query* nya. Untuk menghitung *survival rate*, yakni dengan menulis *query* untuk mengambil data jumlah ikan yang diperpanjang dan jumlah ikan yang dimasukkan ke dalam kolam ketika musim budidaya dimulai, kemudian kedua data tersebut dibagi, kemudian hasil pembagian tersebut dikalikan dengan 100. *Query pipeline* ini digunakan pada API Pond Status yang akan mengembalikan data status kolam termasuk Survival Rate dalam bentuk JSON.

3. Menambahkan Fungsi untuk mengirimkan foto ketika panen

Fungsi pada Lampiran II poin 2.3 akan menyimpan foto ke dalam folder instance/upload, dengan cara mengambil file dari request HTTP dengan nama *field* 'image', kemudian memeriksa apakah ada file yang dikirimkan pada *field*

'image' tersebut, jika ya maka file akan disimpan ke dalam folder instance/upload. Fungsi ini digunakan pada API Panen agar ketika panen dapat mengirimkan foto-foto ikan hasil panen di aplikasi Android.

4. Memperbaiki Fitur Pencatatan Kualitas Air Harian

Fungsi pada Lampiran II poin 2.4 akan mengambil nilai yang dikirimkan oleh request HTTP dengan field 'pH', 'DO', dan 'temperature', kemudian mengubah tipe datanya menjadi *double*, kemudian data tersebut akan disimpan ke dalam tabel Daily Water quality. Fungsi ini digunakan oleh API Daily Water Quality yang dapat mengambil dan mengirimkan data kualitas air harian di aplikasi Android.

5. Memperbaiki Fitur Pencatatan Kematian Ikan

Fungsi pada Lampiran II poin 2.5 akan mengambil nilai yang dikirimkan oleh request HTTP dengan field 'fish_death_amount' di mana list ini berisi list ikan yang mati dan memiliki atribut seed_id, weight, dan amount, kemudian menyimpan data tersebut ke dalam tabel Fish Death dan Fish Grading untuk *tracking* data kematian ikan selama musim budidaya. Fungsi ini digunakan pada API Fish Death dan dapat dipanggil di aplikasi Android.

6. Memperbaiki Fitur Aktivasi Kolam agar dapat menerima data format *double* pada tinggi air

Fungsi pada Lampiran II poin 2.6 akan mengambil nilai yang dikirimkan oleh request HTTP kemudian mengubah nilai pada field 'water_level' menjadi *double* dengan menggunakan float() untuk membungkus variabel yang menyimpan data tinggi air kemudian mengirimkan datanya ke tabel Pond Activation. Fungsi ini digunakan pada API Pond Activation dan dapat dipanggil di aplikasi Android.

Setelah memperbaiki fitur-fitur pada aplikasi versi kedua, peneliti melakukan pengujian *unit testing* yang diuji oleh tim internal *developer*, dan telah mendemokan

aplikasi kepada beberapa pembudidaya ikan air tawar di Jasinga. Pembudidaya pertama yang kami demokan dan ikut menguji aplikasi yakni Pak Ending, seorang petani yang sedang belajar untuk membudidayakan ikan. Ketika selesai kami demokan aplikasi, Pak Ending

4.1.3 Sprint 3

Sprint 3 dilaksanakan pada tanggal 15 Oktober 2023 - 14 November 2023. Detail dari Sprint 3 ini adalah mengerjakan tugas yang ada pada Sprint 3 Backlog di tabel berikut.

Tabel 4.7: Sprint 3 Backlog

No	User Story	Task	Status
1	Sistem Lelang	Memperbarui model <i>database</i> tabel Draft Auction List, Auction List dan Bid List Mengimplementasikan model <i>database</i> fitur Membuat Lelang dalam bentuk Flask API Mengimplementasikan <i>route server</i> untuk fitur Membuat Lelang Men-deploy <i>code backend</i> fitur Membuat Lelang ke server jft.web.id agar dapat diakses melalui Flutter	Done Done Done Done

No	User Story	Task	Status
2	Sistem Multiuser	Mengimplementasikan model <i>database</i> Bid List agar pembudaya dapat saling mengajukan penawaran pada lelang masing-masing Merancang model <i>database</i> tabel Transaction dan Request Topup	Done
3	Sistem Keuangan	Mengimplementasikan model <i>database</i> Transaction dan Request Topup dalam bentuk Flask API Merancang dan mengimplementasikan <i>route server</i> untuk tabel Transaction dan Request Topup Membuat function untuk menghitung saldo	Done

No	User Story	Task	Status
		Men-deploy <i>backend</i> code fitur Transaction ke server jft.web.id agar dapat diakses melalui Flutter	Done

Hasil dari pengerjaan Sprint 3 yakni, pertama perubahan pada model *database* tabel Auction Draft List, Auction List dan Bid List.



Bid	
PK	id: ObjectId()
FK	auction_id : ObjectId()
FK	breeder_id : ObjectId()
	breeder_name : String NOT NULL
	breeder_bid : Int NOT NULL
	breeder_phone_number : String NOT NULL
	is_bid_accepted : Boolean NOT NULL
	created_at : Date NOT NULL
	updated_at : Date NOT NULL

Auction	
PK	id: ObjectId()
FK	breeder_id: ObjectId()
FK	auction_draft : ObjectId()
	auction_name: String NOT NULL
	auction_start_date : Date NOT NULL
	auction_end_date : Date NOT NULL
	total_fish_amount: Int NOT NULL
	total_fish_price: Int NOT NULL
	total_fish_weight: Double NOT NULL
	created_at : Date NOT NULL
	updated_at : Date NOT NULL
	isAuctionFinish : Boolean NOT NULL

Auction Draft	
PK	id: ObjectId()
FK	breeder_id: ObjectId()
FK	harvest_id : ObjectId()
FK	pond_activation_id : ObjectId()
	grading_id : ObjectId()
	fish_name: String NOT NULL
	fish_type : String NOT NULL
	fish_weight: Int NOT NULL
	fish_long: String NOT NULL
	fish_amount : Int NOT NULL
	recommended_price: Int NOT NULL
	breeder_price : Int NOT NULL
	pond_origin : String NOT NULL
	created_at : Date NOT NULL
	updated_at : Date NOT NULL

Gambar 4.4: Skema Database Terbaru untuk tabel Auction Draft List, Auction List dan Bid List.

Field terbaru pada tabel Auction Draft List yakni breeder_id untuk referensi ke ObjectId dari pembudidaya yang melakukan panen dan hendak lelang, kemudian harvest_id untuk referensi ObjectId ke data panen ikan, kemudian pond_activation_id untuk referensi ObjectId ke musim budidaya dari ikan yang dipanen, grading_id yang merujuk ke ObjectId data grading atau pertumbuhan ikan,

fish_name untuk mencatat nama ikan yang dipanen dan hendak dilelang, fish_type untuk mencatat tipe ikan yang dipanen dan hendak dilelang, fish_weight mencatat berat ikan yang dipanen dan hendak dilelang, fish_long mencatat ukuran ikan, fish_amount untuk mencatat jumlah ikan yang dipanen dan hendak dilelang, recommended_price untuk mencatat harga yang direkomendasikan oleh fitur penentuan harga otomatis, breeder_price untuk mencatat harga lelang yang diinginkan oleh pembudidaya, pond_origin untuk mencatat asal kolam, serta terakhir created_at dan updated_at untuk mencatat tanggal pembuatan dan pembaruan ketika data disimpan di *database*

Kemudian, *field* terbaru dari tabel Auction List yang berubah nama menjadi Auction saja, yakni breeder_id untuk referensi pembudidaya yang melakukan lelang, auction_draft yakni *array* yang berisi auction_draft_id dari ikan yang dilelang, kemudian auction_name yang mencatat nama lelang (Paket Lelang A, Lelang Super Murah, dan lain-lain), auction_start_date dan auction_end_date yang mencatat tanggal lelang dimulai dan tanggal lelang berakhir, fish_type untuk mencatat jenis ikan yang akan dilelang, total_fish_amount untuk mencatat total jumlah ikan yang akan dilelang, total_fish_price mencatat total harga lelang, total_fish_weight untuk mencatat total berat ikan yang dilelang, isAuctionFinish untuk menandakan apakah lelang masih berlangsung atau sudah selesai, serta terakhir created_at dan updated_at untuk mencatat tanggal pembuatan dan pembaruan ketika data dikirim ke *database*.

Pembaruan model terakhir yakni pada model tabel Bid List yang juga berubah nama menjadi Bid saja, terdiri dari *field* auction_id untuk referensi ke tabel Auction, breeder_id untuk referensi ke tabel Breeder dan untuk menandakan identitas pembudidaya yang mengajukan penawaran, breeder_name untuk mencatat nama penawar, breeder_bid untuk mencatat jumlah penawaran yang diajukan, breeder_phone_number untuk mencatat nomor telepon dari penawar, is_bid_accepted untuk menandakan apakah penawaran yang diajukan diterima oleh pelelang atau tidak, dan terakhir created_at dan updated_at untuk mencatat tanggal

pembuatan dan pembaruan ketika data dikirimkan ke *database*.

Hasil berikutnya dari penggerjaan Sprint 3 yakni mengimplementasikan model *database* ke dalam bentuk API Flask, mengimplementasikan *route server* dan mendeploy *code backend* ke server jft.web.id:

1. API untuk mengambil data seluruh hasil panen (HTTP Method - GET)

Fungsi pada Lampiran III poin 3.1 akan mengirimkan *request* GET ke server dan akan mengembalikan data dari tabel Harvest untuk dapat diolah dan menampilkan data hasil panen ikan di aplikasi Flutter. Pada fungsi tersebut, *pipeline* berfungsi sebagai *query* di MongoDB untuk mencari data sesuai dengan spesifikasi yang ditentukan, agar data yang didapat sesuai. Apabila data yang dicari ada maka akan mengembalikan respon kode 200 beserta data yang dicari (dalam hal ini, data yang dimaksud yakni data hasil panen). Jika ada kendala seperti data yang dicari tidak ada, maka akan mengembalikan respon kode 400.

2. API untuk mengirim data hasil panen untuk disimpan di tabel Auction Draft (HTTP Method - POST)

Fungsi pada Lampiran III poin 3.2 akan mengirimkan *request* POST ke server dan akan mengirimkan data hasil panen untuk disimpan ke tabel Auction Draft melalui aplikasi Flutter.

Pada fungsi di atas, ada variabel yang menyimpan nilai dari `request.form.get` yang digunakan untuk mengambil data yang dikirimkan oleh Flutter untuk kemudian diolah dan disimpan ke dalam tabel Auction Draft di *database*. Walaupun method fungsi di atas adalah POST, tetapi terdapat juga *query pipeline* untuk mengambil data dari tabel Harvest untuk kemudian diolah dan digunakan untuk menyimpan data ke dalam tabel Auction Draft.

3. API untuk mengambil data dari tabel Auction Draft (HTTP Method - GET)

Fungsi pada Lampiran III poin 3.3 akan mengirimkan *request* GET ke server

dan akan mengambil data dari tabel Auction Draft untuk kemudian ditampilkan di aplikasi Flutter.

4. API untuk mengirim data ke tabel Auction (HTTP Method - POST)

Fungsi pada Lampiran III poin 3.4 akan mengirimkan *request* POST ke server dan akan mengirim data untuk disimpan di tabel Auction agar dapat digunakan oleh aplikasi Flutter.

5. API untuk mengambil data dari tabel Auction (HTTP Method - GET)

Fungsi pada Lampiran III poin 3.5 akan mengirimkan *request* GET ke server dan akan mengambil data dari tabel Auction agar dapat ditampilkan oleh aplikasi Flutter.

6. API untuk mengirim data ke tabel Bid (HTTP Method - POST)

Fungsi pada Lampiran III poin 3.6 akan mengirimkan *request* POST ke server dan akan mengirimkan data ke tabel Bid agar dapat digunakan oleh aplikasi Flutter.

7. API untuk mengambil data dari tabel Bid (HTTP Method - GET)

Fungsi pada Lampiran III poin 3.7 akan mengirimkan *request* GET ke server dan akan mengambil data dari tabel Bid agar dapat ditampilkan oleh aplikasi Flutter di halaman Detail Lelang.

8. API untuk mengirim data ke tabel Request Topup (HTTP Method - POST)

Fungsi pada Lampiran III poin 3.10 akan mengirimkan *request* POST ke server dan akan mengirimkan data ke tabel Request Topup agar dapat digunakan oleh aplikasi Flutter.

9. API untuk mengambil data dari tabel Request Topup (HTTP Method - GET)

Fungsi pada Lampiran III poin 3.11 akan mengirimkan *request* GET ke server dan akan mengambil data dari tabel Request Topup agar dapat ditampilkan oleh Admin di Postman.

10. API untuk menyetujui Request Topup (HTTP Method - PUT) Fungsi pada Lampiran III poin 3.12 akan mengirimkan *request* PUT ke server dan akan mengubah data dari tabel Request Topup ketika Admin menyetujui permintaan topup.
11. API untuk menghitung dan mengembalikan jumlah saldo (HTTP Method - GET) Fungsi pada Lampiran III poin 3.13 akan mengirimkan *request* GET ke server dan akan mengambil data dari tabel Transaction kemudian mengembalikan jumlah saldo yang dimiliki pembudidaya agar dapat ditampilkan oleh aplikasi Flutter.
12. Implementasi Route Server
Ada beberapa perubahan pada *route server*, contohnya yakni *route* untuk membuat *draft* lelang sebelumnya /auction/draft/create sedangkan *route* terbarunya yakni /auction/drafts, lalu untuk membuat lelang *route* sebelumnya yakni /auction/create kemudian diubah menjadi /auctions. Untuk detailnya bisa dilihat pada Lampiran III poin 3.14.
13. Men-deploy *code backend* ke server jft.web.id

```
[root@jft ~]# cd ./var/www/html/fishapi4
[root@jft fishapi4]# ls
--v4
  README.md  exportdatabase  init-flask.sh  pyvenv.cfg
  README-V2-PKM  bin  fishapi4  instance  wsgi.py
  LICENSE  doc  index.wsgi  node_modules  yarn.lock
[root@jft fishapi4]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset
: disabled)
   Active: active (running) since Wed 2024-01-10 12:58:58 WIB; 2 days ago
     Docs: man:httpd(8)
           man:apachectl(8)
   Process: 11934 ExecStop=/bin/kill -WINCH ${MAINPID} (code=exited, status=0/SUC
CESS)
 Main PID: 11949 (httpd)
   Status: "Total requests: 1919; Current requests/sec: 0; Current traffic:  0
B/sec"
```

Gambar 4.5: Server jft.web.id Pasca Deploy Fitur Lelang dan Multiuser

Hasil berikutnya dari penggeraan Sprint 3 yakni mengimplementasikan model *database* Bid List untuk fitur Multiuser, untuk implementasinya dapat dilihat pada poin sebelumnya. Hasil terakhir dari penggeraan Sprint 3 yakni:

1. Merancang model *database* tabel Transaction

Pada tabel Transaction, terdapat *field* breeder_id untuk referensi ke tabel Breeder dan menandakan identitas pembudidaya yang melakukan transaksi, bid_id untuk referensi ke tabel Bid, transaction_type untuk menandakan jenis transaksi, apakah transaksi masuk atau transaksi keluar, transaction_amount untuk mencatat nominal transaksi, desc untuk mencatat deskripsi transaksi, terakhir created_at dan updated_at untuk mencatat tanggal pembuatan dan pembaruan data dikirim ke *database*.

2. API untuk mengirim data ke tabel Transaction (HTTP Method - POST)

Fungsi pada Lampiran III poin 3.8 akan mengirimkan *request* POST ke server dan akan mengirim data untuk disimpan di tabel Transaction agar dapat digunakan oleh aplikasi Flutter.

3. API untuk mengambil data dari tabel Transaction (HTTP Method - GET)

Fungsi pada Lampiran III poin 3.9 akan mengirimkan *request* GET ke server dan akan mengambil data dari tabel Transaction agar dapat ditampilkan oleh aplikasi Flutter.

4. Implementasi Route Server

Untuk detailnya bisa dilihat pada Lampiran III poin 3.14.

5. Men-deploy *code backend* ke server jft.web.id

```
[root@jft ~]# cd ../var/www/html/fishapiv4
[root@jft fishapiv4]# ls
 README.md  exportdatabase  init-flask.sh  pyvenv.cfg
 .gitignore  bin          fishapiv4    instance  wsgi.py
 LICENSE     doc          index.wsgi   node_modules  yarn.lock
 [root@jft fishapiv4]# systemctl status httpd
 ● httpd.service - The Apache HTTP Server
      Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
      Active: active (running) since Wed 2024-01-10 12:58:58 WIB; 2 days ago
        Docs: man:httpd(8)
               man:apachectl(8)
     Process: 11934 ExecStop=/bin/kill -WINCH ${MAINPID} (code=exited, status=0/SUCCESS)
    Main PID: 11949 (httpd)
      Status: "Total requests: 1919; Current requests/sec: 0; Current traffic: 0 B/sec"
```

Gambar 4.6: Server jft.web.id Pasca Deploy Fitur Sistem Keuangan

4.1.4 Sprint 4

Sprint 4 dilaksanakan pada tanggal 15 November 2023 - 14 Desember 2023. Detail dari Sprint 4 ini adalah mengerjakan tugas yang ada pada Sprint 4 Backlog di tabel berikut.

Tabel 4.9: Sprint 4 Backlog

No	User Story	Task	Status
1	Sistem Lelang	<i>Slicing Mockup</i> Fitur Membuat Lelang Membuat desain <i>mockup</i> fitur Detail Lelang, Lelang Grosir, Lelang Saya, Daftar Lelang, Melihat Daftar Penawaran, Menerima Penawaran dan Melihat Riwayat Penawaran <i>Slicing mockup</i> fitur Detail Lelang, Lelang Grosir, Lelang Saya, Daftar Lelang, Melihat Daftar Penawaran, Menerima Penawaran dan Melihat Riwayat Penawaran	Done Done Done Done

No	User Story	Task	Status
2	Sistem Multiuser	Membuat desain fitur Mengajukan Penawaran dan Menerima Penawaran <i>Slicing mockup</i> fitur Mengajukan Penawaran dan Menerima Penawaran Membuat desain tampilan <i>mockup</i> fitur Topup <i>Slicing mockup</i> fitur Topup	Done Done Done Done
3	Sistem Keuangan	Integrasi Maps API ke dalam aplikasi Flutter.	Not Done
4	Maps API		

Hasil dari penggerjaan Sprint 4, yakni pertama melakukan *slicing mockup* fitur Membuat Lelang di Flutter. Ketika melakukan *slicing mockup*, ada perubahan susunan tampilan dan menu *bottom tab bar* di halaman Dashboard dibandingkan iterasi sebelumnya. Pada iterasi saat ini, berikut tampilan perubahannya:

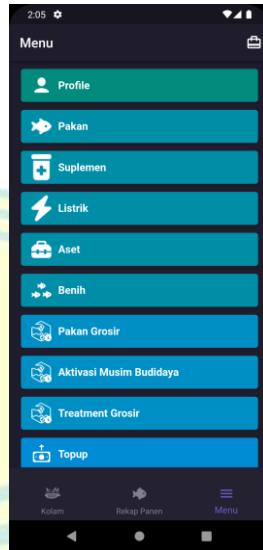
*Mencerdaskan dan
Memartabatkan Bangsa*



Gambar 4.7: Pond Page



Gambar 4.8: Rekap Panen



Gambar 4.9: Menu Page

Sebelumnya, di halaman Dashboard, susunan menu *bottom tab bar* yakni Home Page, Pond Page, dan Profile Page. Pada iterasi saat ini, susunan menu nya berubah menjadi Pond Page, Rekap Panen Page dan Menu Page. Di halaman Pond Page masih berisi Daftar Kolam yang dimiliki pembudidaya, Rekap Panen Page berisi Daftar Panen, Daftar Lelang Grosir dan Daftar Lelang saya, serta tombol *switch* untuk berganti ke halaman Daftar Lelang, sedangkan pada Menu Page berisi tombol-tombol menu, seperti tombol menu Profile Page, Inventaris, Topup, dan menu lainnya. Selain perubahan susunan menu, terdapat juga perubahan pada desain *mockup* fitur Membuat Lelang, seperti Daftar Menu yang sebelumnya muncul ketika menekan tombol menu *hamburger*, kini dipindahkan ke halaman Menu Page di Dashboard. Selain perubahan tampilan Daftar Menu menjadi Menu Page, terdapat perubahan tampilan juga pada alur Membuat Lelang, berikut penampilan terbaru fitur tersebut setelah di-*slicing* ke Flutter:

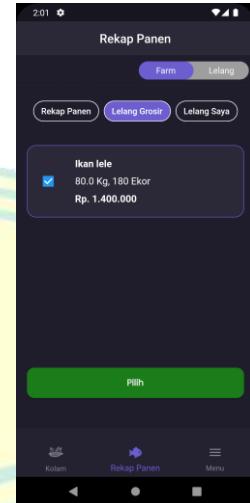
*Mencerdaskan dan
Memartabatkan Bangsa*



Gambar 4.10: Rekap Panen untuk Membuat Lelang Grosir



Gambar 4.11: Membuat Lelang Grosir



Gambar 4.12: Langkah Pertama Membuat Lelang



Gambar 4.13: Langkah Kedua Membuat Lelang



Gambar 4.14: Lelang Saya

Di halaman Rekap Panen, terdapat perubahan informasi yang ditampilkan pada *widget card*, kini menampilkan informasi jenis ikan, FCR, jumlah ikan per kg, asal kolam, durasi musim budidaya dan harga penawaran awal lelang. Untuk membuat lelang, pertama pembudidaya harus meng-klik tombol Lelang pada *widget*

card data panen yang diinginkan (lihat Gambar 4.9), kemudian pembudidaya akan diarahkan ke halaman Konfirmasi Detail Lelang, yang menampilkan *widget card* yang berisi informasi jenis ikan, total berat ikan, jumlah ikan per kilogram, durasi musim budidaya dan harga penawaran lelang. Pembudidaya akan diminta untuk mengisi harga penawaran awal lelang yang diinginkan, sebab harga yang tertera di atas merupakan harga rekomendasi dari fitur penentuan harga yang dikembangkan pada iterasi sebelumnya. Harga yang dimasukkan oleh pembudidaya tidak boleh lebih rendah dari harga yang ditampilkan pada *widget card*. Di bawah kolom *input* harga, akan ditampilkan foto ikan yang panen, dengan catatan pembudidaya mengirimkan foto ketika panen, jika pembudidaya tidak mengirimkan foto, maka akan ditampilkan informasi teks bertuliskan "Tidak Ada Gambar". Setelah selesai mengisi harga penawaran awal yang diinginkan, maka pembudidaya harus meng-klik tombol Konfirmasi untuk mengirimkan data ke *database*.

Di halaman Lelang Grosir, terdapat data hasil panen yang hendak dilelang dan disimpan di tabel *Auction Draft* di *database*. Di halaman ini, pembudidaya dapat melelang satu hasil panen atau membuat paket lelang yang berisi beberapa hasil panen sekaligus dengan cara meng-klik kolom *checkbox* pada hasil panen yang hendak dilelang (lihat Gambar 4.11). Kemudian, untuk membuat lelang, pembudidaya harus meng-klik tombol Pilih, lalu pembudidaya akan diarahkan ke halaman Konfirmasi Detail Lelang. Di halaman ini, terdapat *input* untuk memasukkan nama lelang yang hendak dibuat, seperti Paket Lelang A, Paket Lelang Super Murah, dan lain-lain, kemudian ada informasi tanggal lelang dimulai dan berakhir, di mana tanggal lelang dimulai akan otomatis diatur pada tanggal saat lelang grosir terpilih dibuat dan tanggal lelang berakhir otomatis diatur tepat 2 minggu setelah tanggal lelang dimulai. Kedua tanggal tersebut masih bisa diubah dengan meng-klik *icon button* kalender yang terletak di sebelah kiri tanggal yang ditampilkan (lihat Gambar 4.12). Di bagian bawah halaman ada *widget card* yang berisi informasi jenis ikan, berat dan jumlah ikan, serta harga penawaran awal yang sebelumnya dimasukkan oleh pembudidaya. Apabila pembudidaya mengirimkan

foto ketika panen maka akan ditampilkan pada halaman ini, namun jika tidak maka akan tampil informasi teks bertuliskan "Tidak Ada Gambar". Apabila nama lelang, tanggal dimulai dan berakhir lelang sudah di-*input* oleh pembudidaya, maka pembudidaya harus menekan tombol "Buat Lelang" untuk mengirimkan data ke *database*. Data ikan hasil panen yang telah dilelang dapat dilihat pada halaman Lelang Saya dengan menekan tombol Daftar Lelang Saya di halaman Rekap Panen. Pada halaman Daftar Lelang Saya, terdapat daftar *widget card* yang menampilkan informasi nama lelang, jenis lelang, harga penawaran awal lelang, tanggal lelang dimulai dan tanggal lelang berakhir.

Hasil berikutnya dari penggerjaan Sprint 4 yakni membuat desain *mockup* dan *slicing mockup* untuk halaman Detail Lelang, Lelang Grosir, Lelang Saya, Daftar Lelang, Mengajukan Penawaran, Menerima Penawaran dan Melihat Riwayat Penawaran. Untuk halaman Lelang Grosir dan Lelang Saya bisa dilihat pada Gambar 4.11 dan 4.13. Berikut merupakan hasil *slicing mockup* untuk halaman Detail Lelang, Mengajukan Penawaran, Menerima Penawaran dan Melihat Riwayat Penawaran:



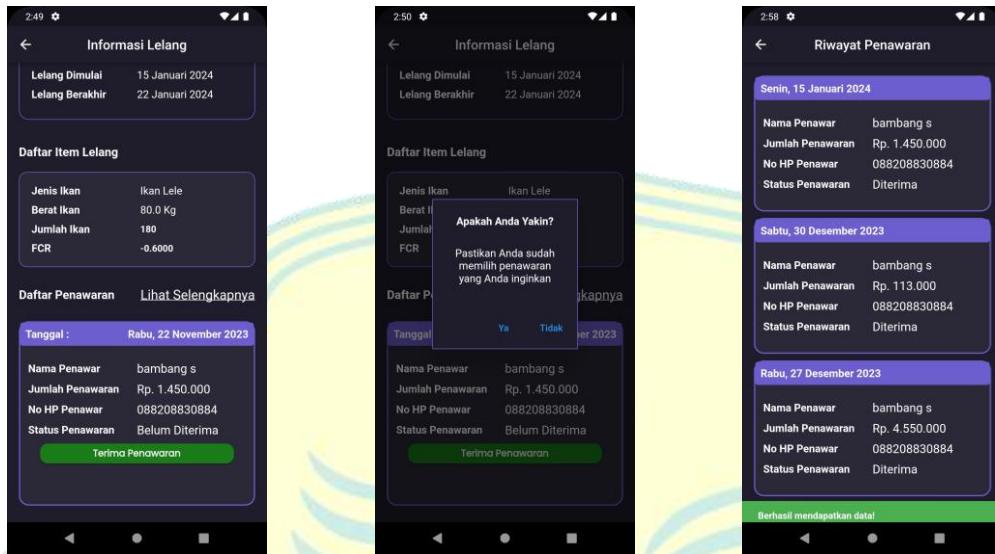
Gambar 4.15: Detail Lelang



Gambar 4.16: Mengajukan Penawaran



Gambar 4.17: Melihat Penawaran



Gambar 4.18: Menerima Penawaran

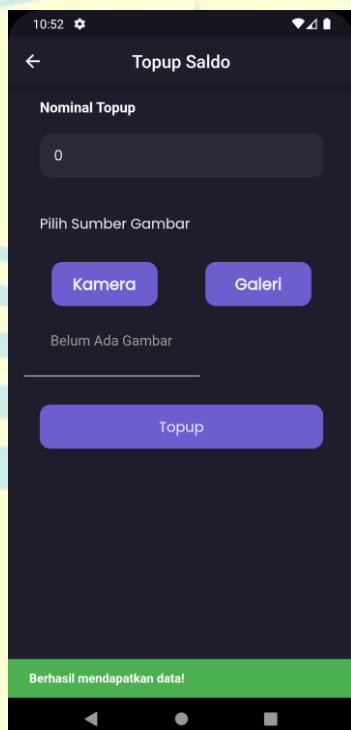
Gambar 4.19: Konfirmasi Menerima Penawaran

Gambar 4.20: Riwayat Penawaran

Pada halaman Detail Lelang, terdapat 2 jenis *widget card* yakni Detail Lelang dan Daftar Item Lelang. *Widget card* Detail Lelang menampilkan informasi lelang seperti nama lelang, harga penawaran awal lelang, lelang dimulai dan lelang berakhir. Sedangkan *widget card* Daftar Item Lelang menampilkan informasi ikan yang tengah dilelang seperti jenis ikan, berat ikan, jumlah ikan dan fcr ikan ketika panen. Lalu, di bawahnya akan menampilkan daftar penawaran yang masuk (apabila ada, seperti pada Gambar 4.16), jika tidak ada maka akan ditampilkan teks "Belum ada penawaran". Tampilan ini adalah tampilan detail lelang apabila pembudidaya yang saat ini sedang *login* merupakan pembudidaya yang melakukan lelang. Jika pembudidaya yang sedang *login* bukan merupakan pembudidaya yang tengah melakukan lelang, maka di bagian bawah *widget card* Daftar Item Lelang akan menampilkan *input* Masukan Jumlah Penawaran (lihat Gambar 4.16). Untuk menerima penawaran, pembudidaya cukup menekan tombol Terima Penawaran pada salah satu penawaran yang diinginkan, kemudian akan muncul *pop-up* untuk mengkonfirmasi penerimaan penawaran, kemudian klik Ya, dan penawaran pun telah diterima (lihat Gambar 4.17 dan Gambar 4.18). Untuk melihat Riwayat Penawaran, halaman tersebut dapat diakses dengan cara klik navigasi ke Menu Page, kemudian

scroll hingga menemukan tombol Riwayat Penawaran, kemudian pembudidaya akan diarahkan ke halaman tersebut.

Hasil terakhir dari penggerjaan Sprint 4 yakni membuat desain *mockup* dan *slicing mockup* untuk fitur Topup, berikut hasilnya:



Gambar 4.21: Topup

Di halaman ini, pembudidaya mengisikan nominal topup dan foto bukti transfer, kemudian klik tombol topup untuk mengirimkan data ke tabel *request topup* di database. Admin perlu menerima *request* yang masuk agar saldo pembudidaya bertambah. Namun, pada penelitian ini, admin menerima permintaan tersebut hanya melalui Postman saja. Untuk implementasi Maps API tidak selesai dikarenakan Maps API dari Google telah berbayar untuk digunakan pada aplikasi Android, sedangkan alternatif gratisnya seperti OpenStreetMaps, peta yang tampil belum ter-update sehingga setelah berdiskusi dengan Dosen Pembimbing dan beberapa pembudidaya, tercapailah kesepakatan bahwa fitur tersebut tidak diperlukan bila peta yang ditampilkan belum ter-update.

4.2 Kesimpulan Sprint

Berikut merupakan rekap hasil penggerjaan seluruh Sprint.

Tabel 4.11: Sprint 1 Backlog

No	User Story	Task	Status
1	Sistem Lelang	Membuat Desain Mockup fitur Membuat Lelang untuk Pembudidaya Pelelang Merancang skema Database untuk fitur Membuat Lelang bagi Pembudidaya Pelelang Membuat tabel yang berisikan route server untuk fitur Membuat Lelang bagi Pembudidaya Pelelang	Done Done Done

No	User Story	Task	Status
		Mengintegrasikan skema Database Lelang yang telah dibuat dengan database pada iterasi kedua	Done

Tabel 4.13: Sprint 2 Backlog

No	User Story	Task	Status
1	Panen	Memperbaiki perhitungan FCR Memperbaiki perhitungan <i>survival rate</i> Menambahkan fungsi untuk mengirimkan foto	Done Done Done
2	Pencatatan Kualitas Air	Memperbaiki fitur pencatatan kualitas air harian	Done
3	Pencatatan Kematian Ikan	Memperbaiki fitur pencatatan kematian ikan	Done

No	User Story	Task	Status
4	Aktivasi Kolam	Memperbaiki fitur aktivasi kolam agar dapat mengirimkan data tinggi air dalam format <i>double</i>	Done

Tabel 4.15: Sprint 3 Backlog

No	User Story	Task	Status
1	Sistem Lelang	Memperbarui model <i>database</i> tabel Draft Auction List, Auction List dan Bid List Mengimplementasikan model <i>database</i> fitur Membuat Lelang dalam bentuk Flask API Mengimplementasikan <i>route server</i> untuk fitur Membuat Lelang Men-deploy <i>code backend</i> fitur Membuat Lelang ke server jft.web.id agar dapat diakses melalui Flutter	Done Done Done Done

No	User Story	Task	Status
2	Sistem Multiuser	Mengimplementasikan model <i>database</i> Bid List agar pembudaya dapat saling mengajukan penawaran pada lelang masing-masing Merancang model <i>database</i> tabel Transaction dan Request Topup	Done
3	Sistem Keuangan	Mengimplementasikan model <i>database</i> Transaction dan Request Topup dalam bentuk Flask API Merancang dan mengimplementasikan <i>route server</i> untuk tabel Transaction dan Request Topup Membuat function untuk menghitung saldo	Done

No	User Story	Task	Status
		Men-deploy <i>code backend</i> fitur Transaction ke server jft.web.id agar dapat diakses melalui Flutter	Done

Tabel 4.17: Sprint 4 Backlog

No	User Story	Task	Status
1	Sistem Lelang	<i>Slicing Mockup</i> Fitur Membuat Lelang Membuat desain <i>mockup</i> fitur Detail Lelang, Lelang Grosir, Lelang Saya, Daftar Lelang, Melihat Daftar Penawaran, Menerima Penawaran dan Melihat Riwayat Penawaran	Done Done

Mencerdaskan dan
Memartabatkan Bangsa

No	User Story	Task	Status
2	Sistem Multiuser	<p><i>Slicing mockup</i> fitur Detail Lelang, Lelang Grosir, Lelang Saya, Daftar Lelang, Melihat Daftar Penawaran, Menerima Penawaran dan Melihat Riwayat Penawaran</p> <p>Membuat desain fitur Mengajukan Penawaran dan Menerima Penawaran</p> <p><i>Slicing mockup</i> fitur Mengajukan Penawaran dan Menerima Penawaran</p>	Done
3	Sistem Keuangan	<p>Membuat desain tampilan <i>mockup</i> fitur Topup</p> <p><i>Slicing mockup</i> fitur Topup</p>	Done
4	Maps API	Integrasi Maps API ke dalam aplikasi Flutter.	Not Done

*Mencerdaskan dan
Memartabatkan Bangsa*

Dapat dilihat dari tabel-tabel di atas, bahwa hampir seluruh Sprint berhasil diwujudkan, kecuali satu Sprint yakni Implementasi Maps API. Peneliti tidak berhasil

mewujudkan Sprint tersebut dikarenakan peneliti menghabiskan banyak waktu untuk memperbaiki aplikasi versi kedua dan mengujinya sehingga tidak memiliki waktu yang cukup untuk mengimplementasikan Maps API ke dalam aplikasi.

Namun, walaupun Sprint tersebut tidak berhasil diwujudkan, namun aplikasi Aqua Breeding versi ketiga yang memperbaiki kekurangan pada versi kedua dan menambahkan fitur sistem lelang, sistem multiuser dan sistem keuangan berhasil diciptakan dan berfungsi dengan sangat baik. Peneliti telah mengujikan masing-masing fitur dengan tim internal *developer* menggunakan metode *Unit-Testing*. Peneliti juga telah mengujikan aplikasi dengan beberapa pembudidaya ikan di Kabupaten Bogor dan telah mendapatkan validasi dari pembudidaya-pembudidaya tersebut, untuk detailnya bisa dilihat transkrip wawancara dengan pembudidaya di Lampiran IV hingga VII.

4.3 Pengujian Sistem

1. *Unit Testing*

Aplikasi versi ketiga telah melalui serangkaian uji coba dengan menggunakan *Unit Testing* yang dilakukan oleh tim pengembang internal. Uji coba ini terdiri dari empat tahap. Pada tahap awal, fokus pengujian aplikasi ditujukan kepada fitur-fitur yang telah diperbaiki dari aplikasi versi kedua selama pengerjaan Sprint 2 pada aplikasi versi ketiga. Rincian hasil dari tahap pertama uji coba ini dapat ditemukan dalam tabel berikut:

Tabel 4.19: Unit testing perbaikan fitur aplikasi versi kedua.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Perhitungan FCR	!		Diterima
Perhitungan <i>Survival Rate</i>	!		Diterima

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Menambahkan fungsi untuk mengirimkan foto ketika panen	!		Diterima
Fitur pencatatan kualitas air harian	!		Diterima
Fitur pencatatan kematian ikan	!		Diterima
Memperbaiki fungsi aktivasi kolam agar menerima data tinggi air bertipe data <i>double</i>	!		Diterima

Setelah pengujian pada tahap pertama selesai dan diterima, maka proses pengujian pun dilanjutkan ke tahap berikutnya. Mulai dari pengujian tahap kedua hingga tahap keempat, pengujian difokuskan pada fitur-fitur baru yang ditambahkan dalam versi ketiga. Fitur yang diujikan pada pengujian tahap kedua yakni fitur lelang. Berikut adalah hasil pengujian fitur lelang:

Tabel 4.20: Unit testing fitur lelang.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Membuat <i>Draft Lelang</i> untuk membuat paket lelang	!		Diterima
Menampilkan data <i>Draft Lelang</i>	!		Diterima
Membuat Lelang sesuai dengan data Draft Lelang yang dipilih	!		Diterima
Menampilkan data Lelang	!		Diterima
Menampilkan informasi detail lelang	!		Diterima
Menampilkan penawaran lelang terpilih	!		Diterima

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Menyaring data Lelang yang tampil sesuai berdasarkan kelurahan/kecamatan/kabupaten yang dipilih	!		Diterima

Pada pengujian tahap ketiga, fitur yang diujikan adalah fitur sistem keuangan.

Berikut adalah hasil pengujian fitur tersebut:

Tabel 4.21: Unit testing fitur sistem keuangan.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Mengirimkan permintaan topup	!		Diterima
Mengajukan penawaran lelang	!		Diterima
Menampilkan riwayat penawaran lelang	!		Diterima
Menampilkan saldo	!		Diterima
Memotong dan mengirimkan saldo sesuai dengan jumlah penawaran yang diajukan (apabila penawaran diterima)	!		Diterima

Pada pengujian tahap akhir, fitur yang diujikan adalah fitur multiuser. Berikut adalah hasil pengujian fitur tersebut:

Tabel 4.22: Unit testing fitur sistem multiuser.

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Login Admin	!		Diterima

Skenario Pengujian	Kesesuaian		Kesimpulan
	sesuai	tidak sesuai	
Admin dapat melihat daftar permintaan topup beserta bukti pembayarannya	!		Diterima
Admin dapat memilih dan menyetujui permintaan topup apabila jumlah permintaan topup dan bukti pembayarannya terbukti valid	!		Diterima

Ketika pengujian tahap 2 dan 3 dilakukan, tim internal *developer* memberikan beberapa catatan saran perubahan terhadap aplikasi. Secara fungsi, fitur-fitur tersebut berhasil berjalan, sehingga tim internal *developer* telah menerima fitur-fitur tersebut, namun memberikan saran untuk mengubah warna di beberapa halaman agar lebih bagus lagi. Kemudian, tim internal *developer* juga menyarankan untuk menambah filter pada halaman Daftar Lelang, untuk mempermudah pembudidaya penawar untuk mencari lelang terdekat dengan domisili.

*Mencerdaskan dan
Memartabatkan Bangsa*

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

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

1. Terbentuknya aplikasi Aqua Breeding versi ketiga dengan fitur sistem multiuser, sistem lelang dan sistem keuangan. Adapun perancangan aplikasi ini dilakukan dengan metode Scrum dimulai dari tahap penyusunan Product Backlog, Sprint Backlog, dan dikerjakan dalam empat Sprint.
2. Hasil pengujian menunjukkan bahwa semua skenario pada *unit testing* berjalan dengan sukses.

5.2 Saran

Adapun saran untuk penelitian selanjutnya adalah:

Berdasarkan diskusi dengan *stakeholder*, perlu menambahkan tampilan Admin di aplikasi Android untuk mempermudah admin dalam menyetujui permintaan topup yang masuk sehingga tidak perlu menggunakan Postman.

*Mencerdaskan dan
Memartabatkan Bangsa*

DAFTAR PUSTAKA

- Alfatih, A. M. (2023). Ekspansi Aplikasi aqua breeding dengan penambahan fitur inventarisasi untuk penentuan harga produk perikanan berbasis mobile. *Program Studi Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Jakarta*.
- Chopde, N. R., & Nichat, M. K. (2013). Landmark Based Shortest Path Detection by Using A* and Haversine Formula. *International Journal of Innovative Research in Computer and Communication Engineering*, Vol. 1, Issue 2, April 2013.
- Dunia, F. A., Wasilah, & Sasongko, C. (2017). *Akuntansi Biaya*. Salemba Empat.
- Hacid, H., & Zighed, A. D. (2005). An Effective Method for Locally NeighborhoodGraphs Updating.
- 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*.
- Hargrave, M. (2019). *Follow-the-Leader Pricing: What it Means, How it Works*. <https://www.investopedia.com/terms/f/follow-the-leader-pricing.asp>
- Isharyanto. (2018). Penetapan Harga Eceran Tertinggi Komoditas Pangan sebagai Hak Konstitusional dalam Perspektif Negara Kesejahteraan. *Jurnal Konstitusi*, 15(3).
- Ju, C., Luo, Q., & Yan, X. (2020). Path Planning Using an Improved A-star Algorithm. *IIth International Conference on Prognostics and System Health Management, PHM-Jinan 2020*.
- Junaidi, M. A., & Maghdahfanti, E. P. (2020). Dampak Pola Kemitraan Melalui E-Commerce Pertanian (Kasus pada Petani Jeruk dengan PT. TaniHub Indonesia di Kabupaten Malang). *Magister Agribisnis, Volume 20 Nomor 2 Juli 2020*.
- Kotler, P., & Armstrong, G. (2018). *Principles of Marketing* (17th ed.). Pearson Education.
- Kumala, Y. C. (2020). *Lelang Indonesia (Serba Serbi Lelang Dan Pelaksanaannya Di Indonesia)* (1st ed.). Deepublish Publisher.
- 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*.
- Munir, R. (2003). *Matematika Diskrit* (2nd ed.). Informatika Bandung.
- Nuraini, I. (2016). *Pengantar Ekonomi Mikro*. UMM Press.
- Pramleonita, M., Yuliani, N., Arizal, R., & Wardoyo, S. E. (2018). Dampak Pola Kemitraan Melalui E-Commerce Pertanian (Kasus pada Petani Jeruk dengan PT. TaniHub Indonesia di Kabupaten Malang). *JURNAL SAINS NATURAL*, Vol. 8 No. 1 (2018).

- Prastowo, N. J., Yanuarti, T., & Depari, Y. (2008). Pengaruh Distribusi Dalam Pembentukan Harga Komoditas dan Implikasinya Terhadap Inflasi. *Working Paper*.
- 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*.
- Rosen, K. H. (2019). *Discrete Mathematics and Its Applications* (8th ed.). McGraw-Hill.
- Siregar, B., Suripto, B., Hapsoro, D., Lo, E. W., Herowati, E., Kusumasari, L., & Nurofik. (2019). *Akuntansi Biaya*. Salemba Empat.
- Smith, A. (1776). *The Wealth of Nations*. W. Strahan; T. Cadell.
- Soemitra, A. (2015). *Bank dan Lembaga Keuangan Syariah* (5th ed.). Prenadamedia Group.
- Tanaya, D. A. (2022). *Manfaat Lelang untuk Kita dan Negara Kita*. <https://www.djkn.kemenkeu.go.id/kpknl-jakarta2/baca-artikel/14785/Manfaat-Lelang-untuk-Kita-dan-Negara-Kita.html>



*Mencerdaskan dan
Memartabatkan Bangsa*

LAMPIRAN I

Object JSON yang dikembalikan oleh endpoint Direction API

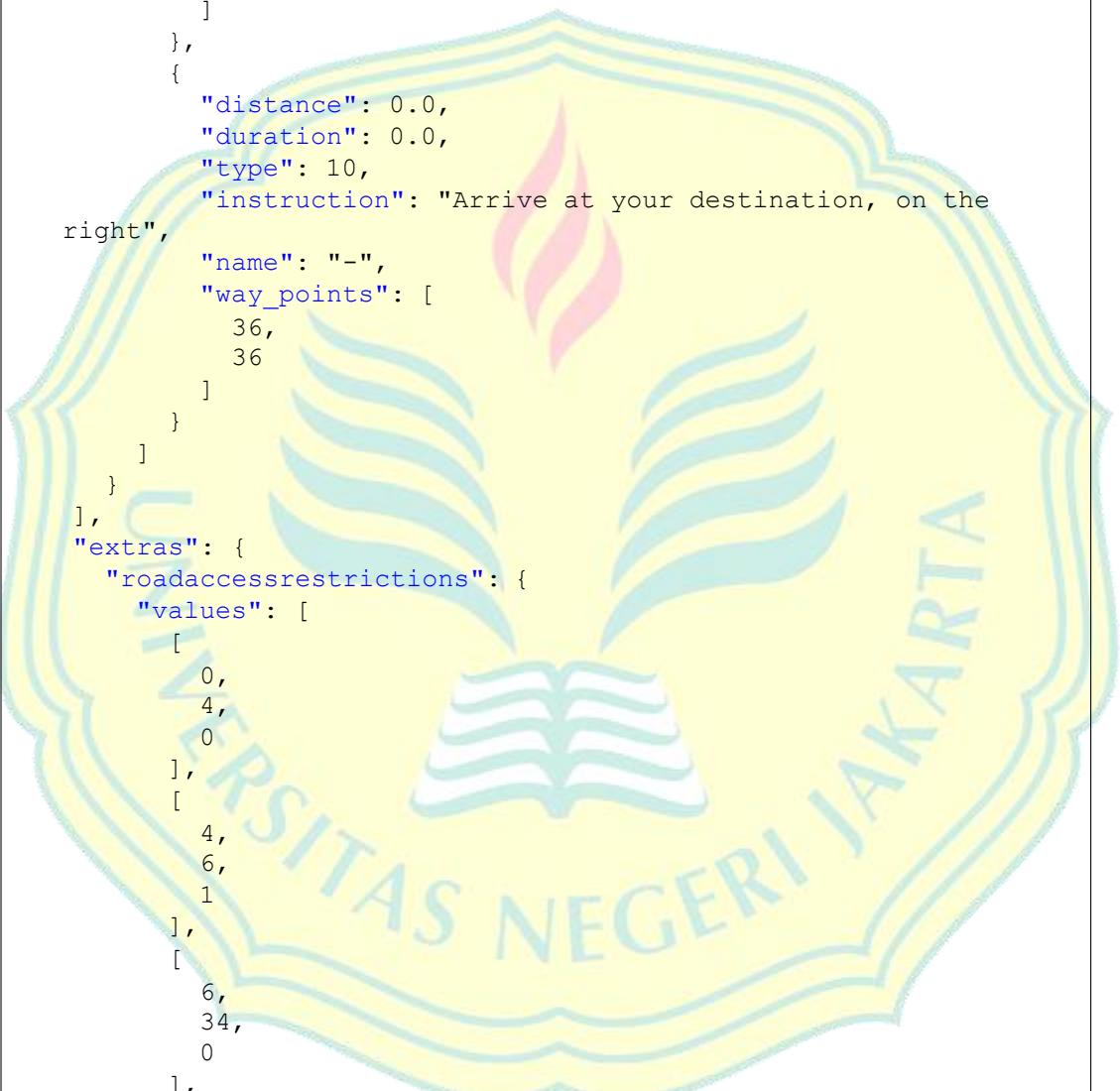
1.1 OpenStreetMaps

```
{
  "type": "FeatureCollection",
  "metadata": {
    "attribution": "openrouteservice.org | OpenStreetMap contributors",
    "service": "routing",
    "timestamp": 1690964556964,
    "query": {
      "coordinates": [
        [
          [
            106.87919451247053,
            -6.194158605350978
          ],
          [
            [
              106.88840350424874,
              -6.193145583778117
            ]
          ],
          "profile": "driving-car",
          "format": "json"
        },
        "engine": {
          "version": "7.1.0",
          "build_date": "2023-07-09T01:31:50Z",
          "graph_date": "2023-07-30T10:09:57Z"
        }
      ],
      "bbox": [
        106.87912,
        -6.196836,
        106.888382,
        -6.193146
      ],
      "features": [
        {
          "bbox": [
            106.87912,
            -6.196836,
            106.888382,
            -6.193146
          ],
          "type": "Feature",
          "properties": {
            "transfers": 0,
            "fare": 0,
            "segments": [

```

```
{  
  "distance": 1725.0,  
  "duration": 234.0,  
  "steps": [  
    {  
      "distance": 162.1,  
      "duration": 38.9,  
      "type": 11,  
      "instruction": "Head south on Jalan Daksina Pati Timur  
II",  
      "name": "Jalan Daksina Pati Timur II",  
      "way_points": [  
        0,  
        4  
      ]  
    },  
    {  
      "distance": 46.3,  
      "duration": 11.1,  
      "type": 1,  
      "instruction": "Turn right onto Jalan Daksina Pati  
Timur A",  
      "name": "Jalan Daksina Pati Timur A",  
      "way_points": [  
        4,  
        6  
      ]  
    },  
    {  
      "distance": 48.2,  
      "duration": 11.6,  
      "type": 0,  
      "instruction": "Turn left onto Jalan Daksina Pati Timur  
I",  
      "name": "Jalan Daksina Pati Timur I",  
      "way_points": [  
        6,  
        7  
      ]  
    },  
    {  
      "distance": 42.8,  
      "duration": 10.3,  
      "type": 1,  
      "instruction": "Turn right onto Jalan Daksina Pati  
Tenggara",  
      "name": "Jalan Daksina Pati Tenggara",  
      "way_points": [  
        7,  
        8  
      ]  
    },  
  ],  
  "time": 15455555555555555  
}
```

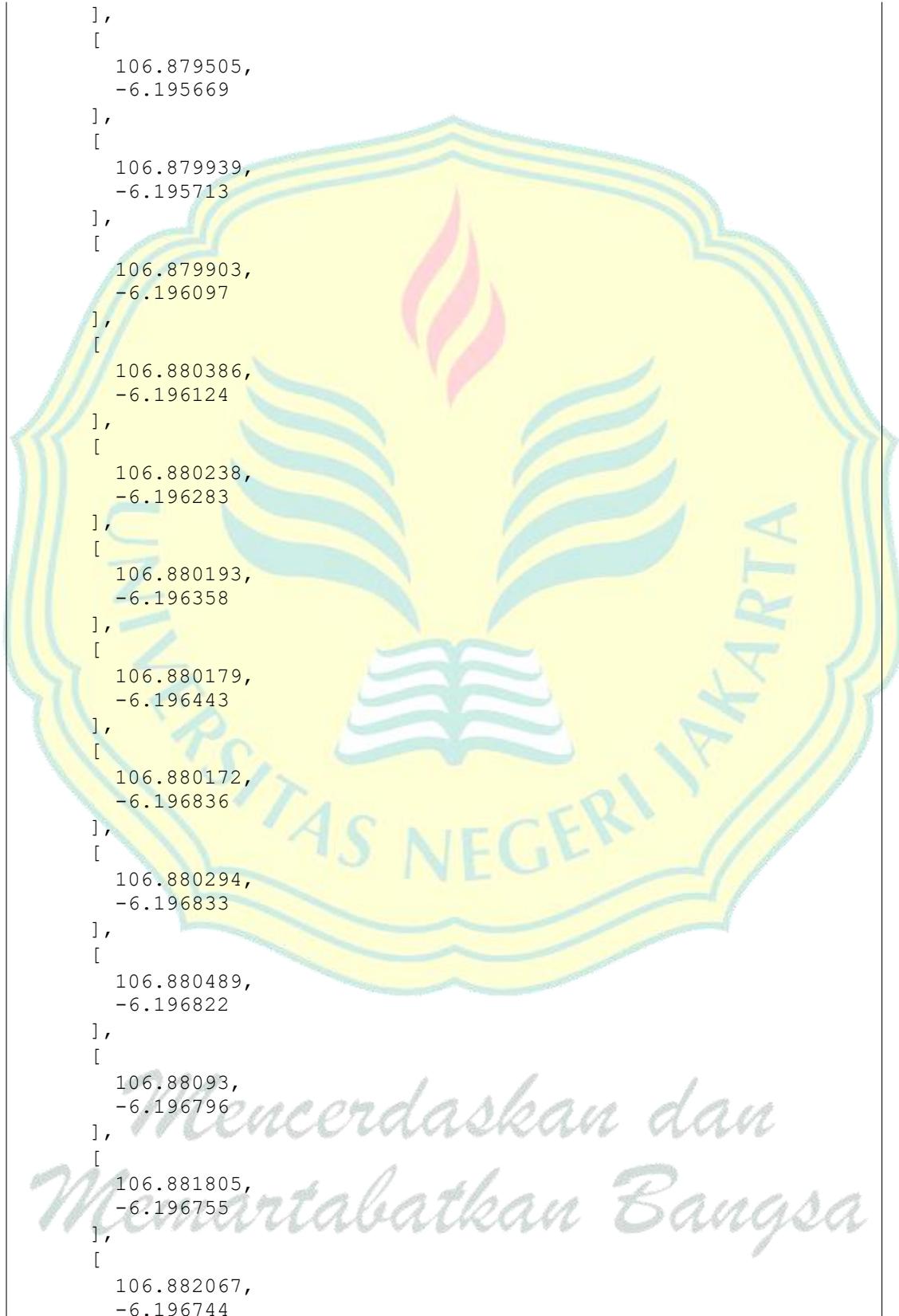
```
{  
    "distance": 140.5,  
    "duration": 44.2,  
    "type": 0,  
    "instruction": "Turn left onto Jalan Daksina Pati  
Tenggara",  
    "name": "Jalan Daksina Pati Tenggara",  
    "way_points": [  
        8,  
        13  
    ]  
},  
{  
    "distance": 649.1,  
    "duration": 57.5,  
    "type": 0,  
    "instruction": "Turn left onto Jalan Rawamangun Muka  
Raya",  
    "name": "Jalan Rawamangun Muka Raya",  
    "way_points": [  
        13,  
        24  
    ]  
},  
{  
    "distance": 366.6,  
    "duration": 32.9,  
    "type": 0,  
    "instruction": "Turn left onto Jalan Balai Pustaka Baru  
",  
    "name": "Jalan Balai Pustaka Baru",  
    "way_points": [  
        24,  
        30  
    ]  
},  
{  
    "distance": 247.9,  
    "duration": 19.8,  
    "type": 1,  
    "instruction": "Turn right onto Jalan Pemuda",  
    "name": "Jalan Pemuda",  
    "way_points": [  
        30,  
        34  
    ]  
},  
{  
    "distance": 21.4,  
    "duration": 7.8,  
    "type": 0,  
    "instruction": "Turn left",  
}
```

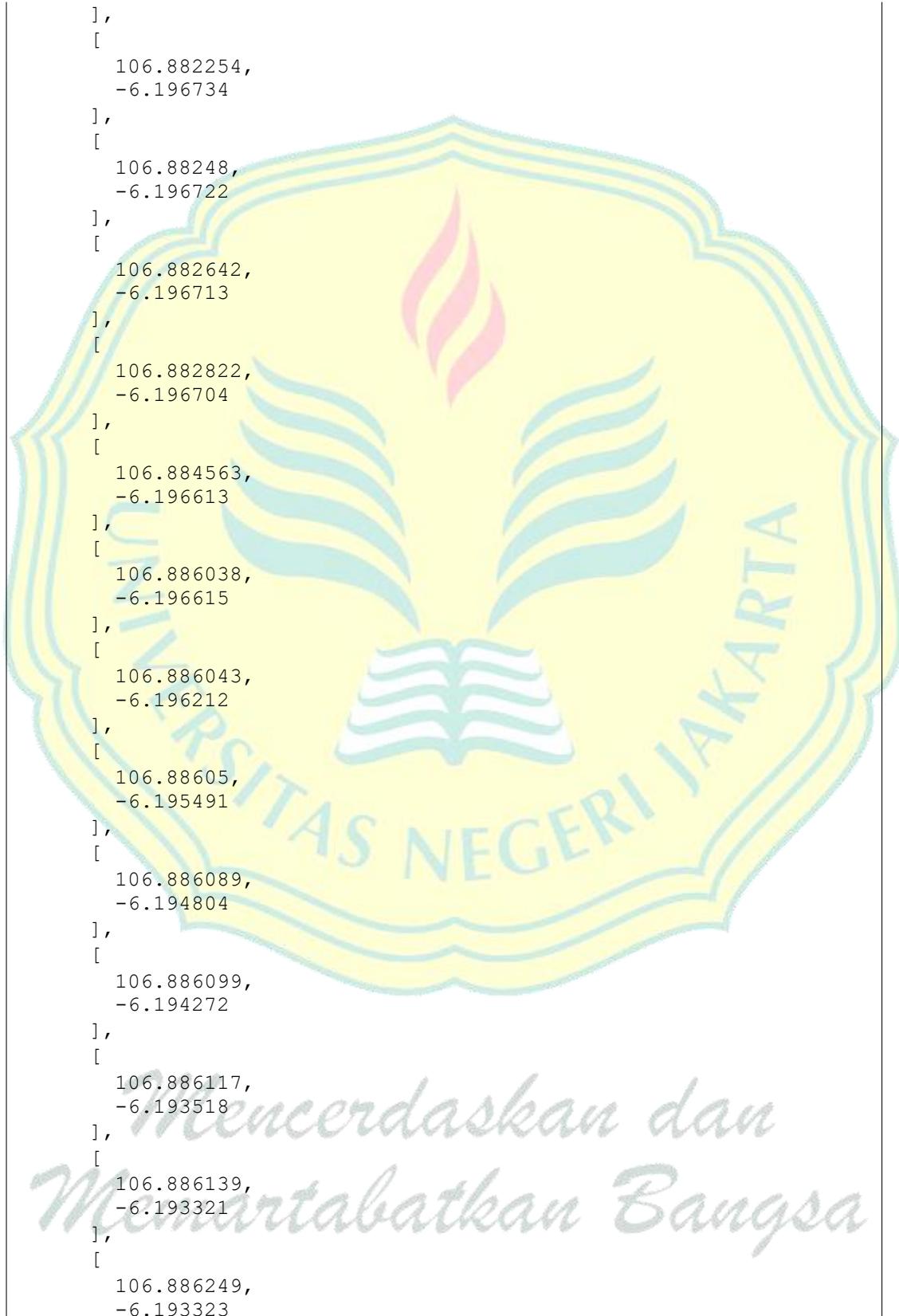


The logo of Universitas Negeri Jakarta (UNJ) is displayed prominently in the background. It features a central yellow shield-shaped emblem. Inside the shield, there is a stylized representation of an open book with horizontal stripes. Above the book, two pink flames rise from either side. The words "VERSITAS NEGERI JAKARTA" are written in blue, curved text along the top inner edge of the shield. Below the shield, a large, faint, handwritten-style text reads "Mencerdaskan dan Meembatkan Bangsa".

```
"name": "-",
"way_points": [
  34,
  36
],
{
  "distance": 0.0,
  "duration": 0.0,
  "type": 10,
  "instruction": "Arrive at your destination, on the
right",
  "name": "-",
  "way_points": [
    36,
    36
  ]
}
],
"extras": {
  "roadaccessrestrictions": {
    "values": [
      [
        0,
        4,
        0
      ],
      [
        4,
        6,
        1
      ],
      [
        6,
        34,
        0
      ],
      [
        34,
        36,
        32
      ],
      {
        "value": 0.0,
        "distance": 1657.3,
        "amount": 96.08
      },
      {
        "value": 0.0,
        "distance": 1657.3,
        "amount": 96.08
      }
    ]
  }
}
```

```
        "value": 1.0,
        "distance": 46.3,
        "amount": 2.68
    },
    {
        "value": 32.0,
        "distance": 21.4,
        "amount": 1.24
    }
]
},
{
    "warnings": [
        {
            "code": 1,
            "message": "There may be restrictions on some roads"
        }
    ],
    "way_points": [
        0,
        36
    ],
    "summary": {
        "distance": 1725.0,
        "duration": 234.0
    }
},
"geometry": {
    "coordinates": [
        [
            [
                106.879217,
                -6.194161
            ],
            [
                106.87912,
                -6.195138
            ],
            [
                106.879128,
                -6.195178
            ],
            [
                106.879173,
                -6.195211
            ],
            [
                106.87952,
                -6.195256
            ],
            [
                106.87951,
                -6.195608
            ]
        ]
    ]
}
```

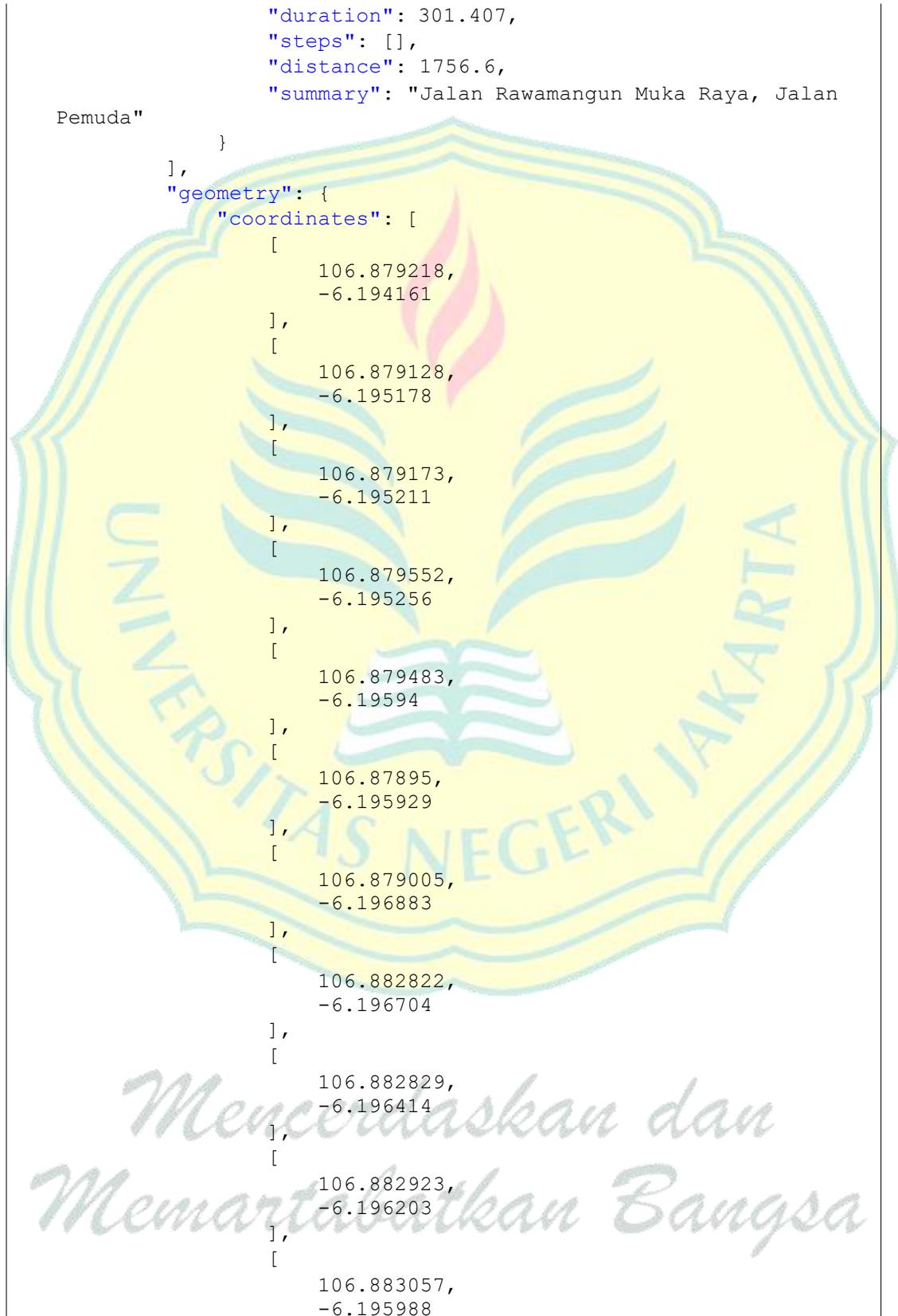


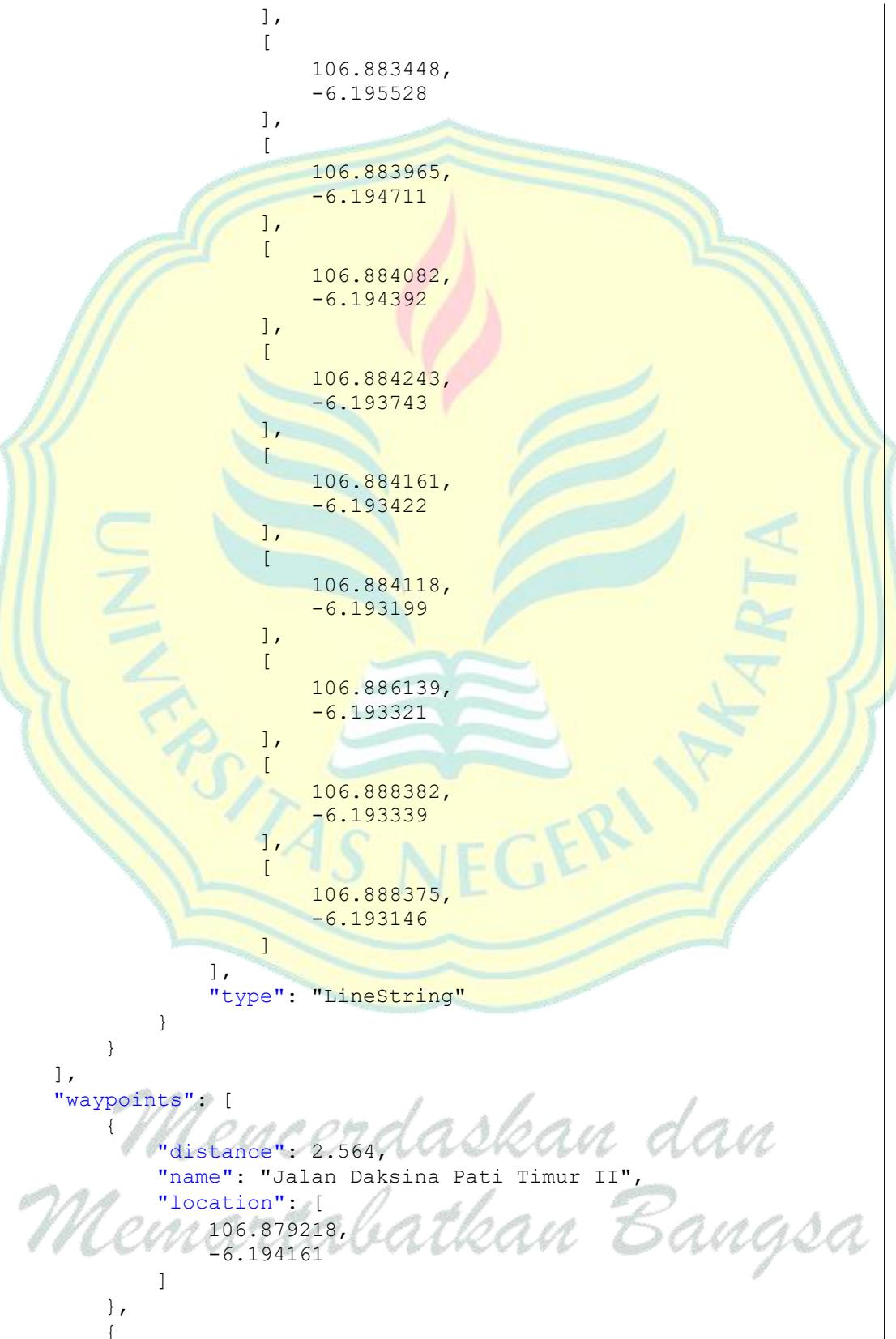


```
],
[
  106.8869,
  -6.193328
],
[
  106.887336,
  -6.193331
],
[
  106.888382,
  -6.193339
],
[
  106.888379,
  -6.193272
],
[
  106.888375,
  -6.193146
]
],
"type": "LineString"
}
]
}
```

1.2 Mapbox

```
{
  "routes": [
    {
      "weight_typical": 835.91,
      "duration_typical": 301.407,
      "weight_name": "auto",
      "weight": 835.91,
      "duration": 301.407,
      "distance": 1756.6,
      "legs": [
        {
          "via_waypoints": [],
          "admins": [
            {
              "iso_3166_1_alpha3": "IDN",
              "iso_3166_1": "ID"
            }
          ],
          "weight_typical": 835.91,
          "duration_typical": 301.407,
          "weight": 835.91,
        }
      ]
    }
  ]
}
```





```

        "distance": 3.123,
        "name": "",
        "location": [
            106.888375,
            -6.193146
        ]
    }
],
"code": "Ok",
"uuid": "vEoi1G5MJvpzXfc7_lUutu22-XfPc2FfQ9fYgEkdkPBu598ptdhKwQ
=="
}

```

1.3 HERE Location Services



```

{
"routes": [
{
    "id": "95536bbd-1780-44ee-b890-ba621dd6ebe4",
    "sections": [
        {
            "id": "84becfe9-7dc5-4b16-a005-ec3627b6da2d",
            "type": "vehicle",
            "departure": {
                "time": "2023-08-02T20:50:21+07:00",
                "place": {
                    "type": "place",
                    "location": {
                        "lat": -6.1941515,
                        "lng": 106.8791412
                    },
                    "originalLocation": {
                        "lat": -6.1941586,
                        "lng": 106.8791945
                    }
                }
            },
            "arrival": {
                "time": "2023-08-02T20:57:56+07:00",
                "place": {
                    "type": "place",
                    "location": {
                        "lat": -6.1931468,
                        "lng": 106.8883877
                    },
                    "originalLocation": {
                        "lat": -6.1931456,
                        "lng": 106.8884035
                    }
                }
            }
        }
    ]
}

```

```
        }
    },
    "polyline": "BGv-
h6LqqS7rG7gC5I3X7BzUTjrBUvR8Buol8BofUoLUgUA4IA8L8B8aAoL
U0PUoaU8QUwHoBgUoBkSwCk1BAkDoBw0BA8BoB0tBTkIAwCAoGAsd0UUUs
EAopBoB8GAokBUkIA8QT8LA8QoBkSUKNoB8LgFAsYA0UTofA8
LTgjBAwMAkD0KnB89BrETrsBUvCwCnB8f8B4DoBoB4DA0UTsOnBk
DvCoB_ib8B57BoE",
    "transport": {
        "mode": "car"
    }
}
]
}
```



Mencerdaskan dan
Memartabatkan Bangsa

LAMPIRAN II

Sprint 2

2.1 Fungsi Untuk Menghitung FCR

```
def _getFcr(pond_activation, total_weight harvested):
    pondact_id=ObjectId(list(pond_activation)[0]['_id'])
    print("pondact_id", pondact_id)
    ## get weight from activation
    pipline=[{
        {
            "$match": {
                "pond_activation_id": ObjectId(pondact_id),
                "event_desc": "ACTIVATION"
            }
        },
    ]
    result = db.aggregate('fish_grading',pipline)
    last_grading_activation = list(result)[0]
    ## get weight from last grading
    pipline=[{
        {
            "$match": {
                "pond_activation_id": ObjectId(pondact_id),
                "event_desc": "GRADING"
            }
        },
        {
            "$sort": {
                "grading_at": -1,
            },
        }
    ]
    result = db.aggregate('fish_grading',pipline)
    last_grading = list(result)
    weightLastGrading = 0.0

    if len(last_grading) is not 0:
        for item in last_grading:
            print("item weight", item['sample_weight'])
            print("item amount", item['sample_amount'])
            weightLastGrading = item['sample_weight'] *
item['sample_amount']
            print("weight after grading", weightLastGrading)
## get date last grading
print("last_grading", last_grading)
date_last_grading = last_grading_activation['grading_at'
,]
```

```

if len(last_grading) is not 0:
    date_last_grading = last_grading[0]['grading_at']
print("date last grading", date_last_grading)
date_now = datetime.datetime.now()
print("date now", date_now)
## get feed amount from last activation
pipeline=[

    {
        "$match": {
            "pond_activation_id": pondact_id,
            "feed_history_time": {"$lte": date_last_grading}
        }
    },
    {
        "$group": {"_id":None,
                    "total_feed_dose_before_last_grading": {"$sum": "$feed_dose"}}
    }
]
result = db.aggregate('feed_history',pipeline)
list_result = list(result)
print("list_result", list_result)
total_feed_dose_before_last_grading = 0.0
if len(list_result) is not 0:
    total_feed_dose_before_last_grading = list_result[0]['total_feed_dose_before_last_grading']
    temp = round(total_feed_dose_before_last_grading, 3)
else:
    total_feed_dose_before_last_grading = temp
print("total_feed_dose_before_last_grading",
total_feed_dose_before_last_grading)
## get feed amount now
pipeline=[

    {
        "$match": {
            "pond_activation_id": ObjectId(pondact_id),
            "feed_history_time": {"$lt": date_now}
        }
    },
]
result = db.aggregate('feed_history',pipeline)
list_result = list(result)
total_feed_dose_now = 0
for item in list_result:
    print("item list feed dose", item['feed_dose'])
    total_feed_dose_now+=item['feed_dose']
temp = round(total_feed_dose_now, 3)
total_feed_dose_now = temp

```

```

        print("total_feed_dose_now", total_feed_dose_now)
        total_fish_weight_last_activation =
_getTotalWeightFromListFishLastGrading(last_grading_activation['
fish'])
        total_fish_weight_now = total_weight_harvested
        print("total fish weight last activation",
total_fish_weight_last_activation)
        print("total fish weight now", total_fish_weight_now)
        diffFeed = float(total_feed_dose_now) - float(
total_feed_dose_before_last_grading)
        if diffFeed == 0.0:
            diffFeed = total_feed_dose_now
        diff = (float(total_fish_weight_now) + float(
weightLastGrading)) - float(total_fish_weight_last_activation)
        if diff == 0.0:
            diff = float(total_fish_weight_now)
        print("diff", diff)
        print("diffFeed", diffFeed)
        fcr = float(diffFeed) / float(diff)
        print("fcr", abs(round(fcr, 3)))
        return abs(round(fcr, 3))
    
```

2.2 Query Pipeline untuk Menghitung Survival Rate

```

pipline = [
{'$match': {'$expr': {'$eq': ['$id', {'$toObjectId': pond_id}]}}, {
    "$lookup": {
        "from": "pond_activation",
        "let": {
            "pondid": "$_id",
        },
        "pipeline": [
            {
                "$match": {
                    "$expr": {
                        "$and": [
                            {
                                "$eq": ["$pond_id", "$$pondid"],
                            }
                        ],
                    }
                },
            },
            {
                "$sort": {
                    "activated_at": -1,
                },
            }
        ]
    }
}
] 
```

```
},
{
  "$lookup": {
    "from": "fish_grading",
    "let": {
      "pond_activation_id": "$_id",
    },
    "pipeline": [
      {
        "$match": {
          "$expr": {
            "$eq": [
              "$pond_activation_id",
              "$$pond_activation_id",
            ],
          },
          "event_desc": "ACTIVATION", # Menambahkan filter
untuk event_desc "ACTIVATION"
        },
      },
      {
        "$sort": {
          "_id": -1,
        },
      },
      {
        "as": "list_grading",
      },
    ],
  },
  {
    "$lookup": {
      "from": "fish_death",
      "let": {
        "pond_activation_id": "$_id",
      },
      "pipeline": [
        {
          "$match": {
            "$expr": {
              "$eq": [
                "$pond_activation_id",
                "$$pond_activation_id",
              ],
            },
          },
        },
        {
          "$sort": {
            "_id": -1,
          },
        },
      ],
    },
  },
}
```

```
        "as": "list_death",
    },
},
{
    "$lookup": {
        "from": "fish_harvested",
        "let": {
            "pond_activation_id": "$_id",
        },
        "pipeline": [
            {
                "$match": {
                    "$expr": {
                        "$eq": [
                            "$pond_activation_id",
                            "$$pond_activation_id",
                        ],
                    },
                },
                {
                    "$sort": {
                        "_id": -1,
                    },
                },
            ],
            "as": "list_fish_harvested",
        },
    },
    {
        "$addFields": {
            "first_fish_harvested": {
                "$first": "$list_fish_harvested",
            },
        },
    },
    {
        "$addFields": {
            "fish_harvested": "$first_fish_harvested.fish",
            "total_fish_harvested": "$first_fish_harvested.
total_fish_harvested",
            "total_weight_harvested": "$first_fish_harvested.
total_weight_harvested",
        },
    },
    {
        "$project": {
            "list_fish_harvested": 0,
        },
    },
    {
        "$lookup": {
```

```
"from": "feed_history",
"let": {
    "pond_activation_id": "$_id",
},
"pipeline": [
    {
        "$match": {
            "$expr": {
                "$and": [
                    {
                        "$eq": [
                            "$pond_activation_id",
                            "$$pond_activation_id",
                        ],
                    },
                    {
                        "$ne": [
                            "$list_grading.fish",
                            null
                        ]
                    }
                ],
                "as": "feed_history"
            }
        }
    },
    {
        "$addFields": {
            "fish_live": {
                "$first": "$list_grading.fish",
            },
            "fish_amount_death": {
                "$cond": {
                    "if": {
                        "$ne": ["$list_death", None],
                    },
                    "then": {
                        "$sum": "$list_death.amount",
                    },
                    "else": 0,
                },
            },
            "sample_weight": {
                "$first": "$list_grading.sample_weight",
            },
            "sample_amount": {
                "$first": "$list_grading.sample_amount",
            },
            "fish_death": [],
            "fish_harvested": [],
            "fish_stock": [],
            "fcr": {
                "$first": "$list_grading.fcr",
            },
            "fcr_update": {
                "$set": {
                    "fish_harvested": [
                        {
                            "pond_activation_id": "$_id",
                            "amount": 0
                        }
                    ],
                    "fish_stock": [
                        {
                            "pond_activation_id": "$_id",
                            "amount": 0
                        }
                    ],
                    "fcr": {
                        "fish_harvested": [
                            {
                                "pond_activation_id": "$_id",
                                "amount": 0
                            }
                        ],
                        "fish_stock": [
                            {
                                "pond_activation_id": "$_id",
                                "amount": 0
                            }
                        ]
                    }
                }
            }
        }
    }
]
```

```
        "$first": "$list_grading.created_at",
    },
    "last_feed_dose": {
        "$first": "$feed_history.created_at",
    },
    "feed_dose": {
        "$sum": "$feed_history.feed_dose",
    },
},
{
    "$addFields": {
        "fish_category": {"$first": "$fish_live.category"}
    }
},
{
    "$addFields": {
        "fish_activation_amount": {"$sum": "$fish_live.amount"},
        "fish_activation_weight": "$sample_weight",
        "total_fish": {
            "$cond": {
                "if": {
                    "$and": [
                        {
                            "$ne": ["$fish_amount_death", None]
                        },
                        {
                            "$ne": ["$fish_amount_death", 0]
                        }
                    ]
                },
                "then": {
                    "$subtract": [
                        {
                            "$sum": "$fish_live.amount"
                        },
                        "$fish_amount_death"
                    ]
                },
                "else": {
                    "$sum": "$fish_live.amount"
                }
            }
        }
    },
    "total_weight": {
        "$cond": {
            "if": {
                "$and": [
                    {
                        "$gt": ["$sample_amount", 0],
                    },
                    {
                        "gt": ["$fish_activation_weight", 0]
                    }
                ]
            }
        }
    }
}
```

```
        "$gt": ["$sample_weight", 0],
    },
],
},
"then": {
    "$multiply": [
        {
            "$sum": "$fish_live.amount",
        },
        {
            "$divide": [
                {
                    "$toDouble": "$sample_weight",
                },
                {
                    "$toDouble": "$sample_amount",
                },
            ],
        },
        {
            "$pow": [
                {
                    "$divide": [
                        {
                            "$sum": "$first_fish_harvested.
total_fish harvested",
                        },
                        0,
                    ],
                },
                0,
            ],
        },
        {
            "$multiply": [
                {
                    "$divide": [
                        {
                            "$sum": "$first_fish_harvested.
total_fish harvested",
                        },
                        {
                            "$sum": "$total_fish_added",
                        },
                    ],
                },
                100,
            ],
        },
    ],
},
```

```
        ],
    },
    "weight_growth": {
        "$subtract": [
            {
                "$multiply": [
                    {
                        "$sum": "$fish_live.amount",
                    },
                    {
                        "$cond": {
                            "if": {
                                "$and": [
                                    {
                                        "$gt": ["$sample_amount", 0],
                                    },
                                    {
                                        "$gt": ["$sample_weight", 0],
                                    }
                                ],
                            },
                            "then": {
                                "$divide": [
                                    {
                                        "$toDouble": "$sample_weight",
                                    },
                                    {
                                        "$toDouble": "$sample_amount",
                                    }
                                ],
                            },
                            "else": 0,
                        },
                    },
                ],
            },
            {
                "$sum": "$fish_live.weight",
            },
        ],
    },
    {
        "$project": {
            "pond_id": 0,
            "feed_history": 0,
            "feed_type_id": 0,
            "list_grading": 0,
            "created_at": 0,
            "updated_at": 0,
        },
    },
}
```

```

        },
    ],
    "as": "pond_activation_list",
},
{
"$addFields": {
    "total_activation": {
        "$size": "$pond_activation_list",
    },
    "pond_activation_list": "$pond_activation_list",
},
{
"$project": {
    "location": 0,
    "shape": 0,
    "material": 0,
    "length": 0,
    "width": 0,
    "diameter": 0,
    "height": 0,
    "image_name": 0,
    "updated_at": 0,
    "created_at": 0,
},
}
]
}

```

2.3 Fungsi untuk Menyimpan Foto Ketika Panen

```

file = request.files['image']
if 'image' not in request.files:
    return {"message": "No file detected"}, 400
if file.filename == '':
    return {"message": "No image selected for
uploading"}, 400
if file and allowed_file(file.filename):
    filename = secure_filename(file.filename)
    path = os.path.join(current_app.
instance_path, current_app.config['UPLOAD_DIR'])
    image_path = os.path.join(path, filename)
    file.save(image_path)

```

2.4 Fungsi untuk Menyimpan Data Kualitas Air Harian (pH, DO dan Suhu Air)

```

class DailyWaterQualitysApi(Resource):
    def post(self):
        try:
            pond_id = request.form.get("pond_id", None)
            pond = db.aggregate('pond', [{"$match": {"_id": ObjectId(pond_id)}}])
            if list(pond)[0]['isActive'] is False:
                response = {"message": "pond is not active"}
                response = json.dumps(response, default=str)
                return Response(response, mimetype="application/json", status=400)
            pipeline = [
                {"$match": {"pond_id": ObjectId(pond_id), "isFinish": False}},
                {"$sort": {"activated_at": -1} },
                {"$limit": 1 }
            ]
            pond_activation = db.aggregate('pond_activation', pipeline)
            pond_activation_id = list(pond_activation)[0]['_id']
            print("pond_activation_id", pond_activation_id)
            body = {
                "pond_id": ObjectId(pond_id),
                "pond_activation_id": ObjectId(pond_activation_id),
                "ph": float(request.form.get("ph", None)),
                "do": float(request.form.get("do", None)),
                "temperature": float(request.form.get("temperature", None)),
                "dailywater_at": request.form.get("dailywater_at", datetime.datetime.now()),
                'created_at' : request.form.get("dailywater_at", datetime.datetime.now()),
                'updated_at' : datetime.datetime.now()
            }
            collection = db.get_collection('daily_water_quality')
            id = collection.insert_one(body)
            response = {
                "message": "success add data daily water quality",
                "id": id.inserted_id
            }
            response = json.dumps(response, default=str)
            return Response(response, mimetype="application/json", status=200)
        except Exception as e:
            response = {"message": str(e)}
            response = json.dumps(response, default=str)
            return Response(response, mimetype="application/json", status=400)

```

2.5 Fungsi untuk Menyimpan Data Kematian Ikan

```

class FishDeathsApi(Resource):
    def post(self):
        try:
            pond_id = request.form.get("pond_id",
None)
            # pond = Pond.objects.get(id=pond_id)
            pond = db.aggregate('pond',[{"$match": {"_id":ObjectId(pond_id)}}])
            if list(pond)[0]['isActive'] == False:
                response = {"message": "status pond
is not active"}
            response = json.dumps(response,
default=str)
            return Response(response, mimetype=
"application/json", status=400)
            pipeline = [
                { "$match": { "pond_id": ObjectId(
pond_id), "isFinish": False } },
                { "$sort": { "activated_at": -1 } },
                { "$limit": 1 }
            ]
            result = db.aggregate('pond_activation
',pipeline)
            pond_activation = list(result)[0]
            pond_activation_id = pond_activation['
_id']
            print("pond_activation",
pond_activation)

            try:
                file = request.files['image', None]
                if not allowed_file(file.filename):
                    response = {"message": "file
type not allowed"}
                default=str)
                response = json.dumps(response,
mimetype="application/json", status=400)
                filename = secure_filename(file.
filename)
                filename = pad_timestamp(filename)
                path = os.path.join(current_app.
instance_path,

```

```
current_app.  
config['UPLOAD_DIR'])  
  
        try:  
            os.makedirs(path)  
        except OSError:  
            pass  
        filepath = os.path.join(path,  
filename)  
  
        file.save(filepath)  
    except:  
        filename = "default.jpg"  
fish_death_amount = request.form.get("fish_death_amount", "[]")  
fish_death_amount = json.loads(fish_death_amount)  
  
if len(fish_death_amount) < 1:  
    response = {"message": "There is no fish"}  
  
response = json.dumps(response,  
mimetype="application/json", status=400)  
  
for item in fish_death_amount:  
    # Convert 'weight' to float  
    item['weight'] = float(item['weight'])  
    item['amount'] = float(item['amount'])  
  
    # Convert 'seed_id' to ObjectId  
    item['seed_id'] = ObjectId(item['seed_id'])  
  
    print("fishess", fish_death_amount)  
    print("pond_activations",  
pond_activation)  
  
    fishGradingData = {  
        "pond_id" : ObjectId(pond_id),  
        "pond_activation_id" : ObjectId(  
pond_activation_id),  
        "event" : pond_activation,  
        "event_desc" : 'DEATH',  
        "fish" : fish_death_amount,  
        "fcr" : 0,  
        "sample_amount": int(fish_death_amount[0]["amount"])),  
        "sample_weight" : 0,  
        "grading_at": datetime.datetime.now,  
        "created_at": datetime.datetime.now  
    }  
()  
()
```

```

        collection = db.get_collection('
fish_grading')
fishGrading =collection.insert_one(
print("success add fish grading")
body = {
    "pond_id": ObjectId(pond_id),
    "pond_activation_id": ObjectId(
pond_activation_id),
    ".inserted_id",
[0]['amount']),
diagnosis", None),
collection_name)
fishdeath", "id" : id}
=id
=application/json", status=200)
except Exception as e:
    response = {"message": str(e)}
response = json.dumps(response, default
return Response(response, mimetype="

application/json", status=400)

```

2.6 Fungsi agar Aktivasi Kolam dapat Menerima Tipe Data Double pada Ketinggian Air

```

pondActivationData = {
"pond_id" : ObjectId(pond_id),
"isFinish" : False,
"fish_type" : fish[0]["type"],
"fish_category" : args.fish_category,
"total_fish_added" : amount,
"water_level" : float(args.water_level),
}
active_at = request.form.get('active_at')

```

```
print("active_at", active_at)
if active_at != '':
    pondActivationData['activated_at'] = datetime.datetime.strptime(
        active_at, "%Y-%m-%dT%H:%M:%S.%f %z")
    pondActivationData['created_at'] = datetime.datetime.strptime(
        active_at, "%Y-%m-%dT%H:%M:%S.%f %z")
    pondActivationData['updated_at'] = datetime.datetime.strptime(
        active_at, "%Y-%m-%dT%H:%M:%S.%f %z")

else :
    three_months_ago = datetime.datetime.now() - datetime.timedelta(
        days=3 * 30)
    pondActivationData['activated_at'] = three_months_ago
    pondActivationData['created_at'] = three_months_ago
    pondActivationData['updated_at'] = three_months_ago

collection = db.get_collection(collection_name)
pondActivation = collection.insert_one(pondActivationData)
```



*Mencerdaskan dan
Memartabatkan Bangsa*

LAMPIRAN III

Sprint 3

3.1 Fungsi untuk Mengambil Seluruh Data Hasil Panen

```
class HarvestApi(Resource):
    def get(self, pond_activation_id):
        try:
            pipeline = [
                {
                    "$match": {
                        "pond_activation_id": ObjectId(pond_activation_id),
                    },
                },
                {
                    "$lookup": {
                        "from": "fish_grading",
                        "localField": "grading_id",
                        "foreignField": "_id",
                        "as": "grading",
                    },
                },
                {"$unwind": {"path": "$grading", "includeArrayIndex": "string"}},
                {"$unwind": {"path": "$grading.fish", "includeArrayIndex": "string"}},
                {
                    "$project": {
                        "fish_type": "$grading.fish.type",
                        "fish_weight": "$grading.fish.weight",
                        "fish_size": "$grading.fish.size",
                        "fish_amount": "$grading.fish.amount",
                        "fish_pond_origin": "$grading.fish.pond_id",
                        "fish_id": 1,
                        "suplemen_id": 1,
                        "feedId": 1,
                        "feed_price": 1,
                        "suplemen_price": 1,
                        "fish_price": 1,
                        "price": {"$sum": [
                            "$fish_price", "$suplemen_price", "$feed_price"]
                        },
                    },
                },
                {
                    "$lookup": {
```

```
        "from": "suplemen_inventory",
        "localField": "suplemen_id",
        "foreignField": "_id",
        "as": "suplemen",
      },
    },
  {
    "$addFields": {
      "suplemen": {
        "$cond": {"if": {"$eq": ["$suplemen", []]}, "then": [{"$set": {"suplemen": null}}], "else": "$suplemen"}
      }
    }
  },
  {
    "$unwind": {"path": "$suplemen", "includeArrayIndex": "string"}},
  {
    "$lookup": {
      "from": "feed_inventory",
      "localField": "feedId",
      "foreignField": "_id",
      "as": "feed"
    }
  },
  {
    "$addFields": {
      "feed": {
        "$cond": {"if": {"$eq": ["$feed", []]}, "then": [{"$set": {"feed": null}}], "else": "$feed"}
      }
    }
  },
  {
    "$unwind": {"path": "$feed", "includeArrayIndex": "string"}},
  {
    "$lookup": {
      "from": "fish",
      "localField": "fish_id",
      "foreignField": "_id",
      "as": "fish"
    }
  },
  {
    "$addFields": {
      "fish": {
        "$cond": {"if": {"$eq": ["$fish", []]}, "then": [{"$set": {"fish": null}}], "else": "$fish"}
      }
    }
  },
  {
    "$unwind": {"path": "$fish", "includeArrayIndex": "string"}},
```

```

    {
        "$lookup": {
            "from": "pond",
            "localField": "fish_pond_origin",
            "foreignField": "_id",
            "as": "pond",
        },
        {"$unwind": {"path": "$pond", "includeArrayIndex": "string"}},
        {
            "$addFields": {
                "pond_name": "$pond.alias",
                "suplemen_name": "$suplemen.name",
                "feed_name": "$feed.brand_name",
                "fish_name": "$fish.brand_name",
                "total_price": {"$sum": ["$fish_price", "$feed_price", "$suplemen_price"]},
            },
        },
        {
            "$project": {
                "pond": 0,
                "string": 0,
                "suplemen": 0,
                "feed": 0,
                "fish": 0,
                "pond": 0,
            },
        },
    ]
}

harvest = db.aggregate(collection_name, pipeline)
list_harvest = list(harvest)
response = json.dumps(list_harvest[0], default=str)
return Response(response, mimetype="application/json", status=200)
except Exception as e:
    response = {"message": str(e)}
    response = json.dumps(response, default=str)
    return Response(response, mimetype="application/json", status=400)

```

3.2 Fungsi untuk Mengirimkan Data ke Tabel Auction Draft

```
auction_draft_collection_name = 'auction_draft'
```

```

class AuctionDraftsApi(Resource):
    @jwt_required()
    def post(self):
        try:
            current_user = get_jwt_identity()
            id = str(current_user['id'])
            breeder_id = ObjectId(id)
            harvest_id_form = request.form.get("harvest_id", None)
            price = request.form.get("price", None)
            harvest_pipeline = [
                {
                    "$match": {
                        "_id": ObjectId(
                            harvest_id_form),
                    },
                    {
                        "$lookup": {
                            "from": "fish_grading",
                            "localField": "grading_id",
                            "foreignField": "_id",
                            "as": "grading",
                        },
                        {
                            "$unwind": {"path": "$grading", "includeArrayIndex": "string"}, },
                            {"$unwind": {"path": "$grading.fish", "includeArrayIndex": "string"}}, {
                                "$project": {
                                    "fish_type": "$grading.fish.type",
                                    "fish_weight": "$grading.fish.weight",
                                    "fish_size": "$grading.fish.size",
                                    "fish_amount": "$grading.fish.amount",
                                    "pond_id": "$grading.pond_id",
                                    "pond_origin": "fish_id": 1,
                                    "suplemen_id": 1,
                                    "feedId": 1,
                                    "feed_price": 1,
                                    "suplemen_price": 1,
                                    "fish_price": 1,
                                    "grading_id": 1,
                                    "pond_activation_id": 1,
                                    "fish_harvested_id": 1,
                                }
                            }
                        ]
                    ],
                    "allowDiskUse": true
                }
            ]
        except Exception as e:
            return {"error": str(e)}, 500
        else:
            return {"message": "Success", "data": data}, 201
    
```

```

        "price": {"$sum": [
    $fish_price", "$suplemen_price", "$feed_price"]},
    },
    {
        "$lookup": {
            "from": "suplemen_inventory",
            "localField": "suplemen_id",
            "foreignField": "_id",
            "as": "suplemen",
        },
        {
            "$addFields": {
                "suplemen": {
                    "$cond": {"if": {"$eq": ["$suplemen", []]}, "then": [{"$set": {"suplemen": null}}], "else": "$suplemen"}
                }
            },
            "$unwind": {"path": "$suplemen", "includeArrayIndex": "string"}}
        },
        {
            "$lookup": {
                "from": "feed_inventory",
                "localField": "feedId",
                "foreignField": "_id",
                "as": "feed",
            },
            {
                "$addFields": {
                    "feed": {
                        "$cond": {"if": {"$eq": ["$feed", []]}, "then": [{"$set": {"feed": null}}], "else": "$feed"}
                    }
                },
                "$unwind": {"path": "$feed", "includeArrayIndex": "string"}}
            },
            {
                "$lookup": {
                    "from": "fish",
                    "localField": "fish_id",
                    "foreignField": "_id",
                    "as": "fish",
                },
            },
        }
    }
}

```



```
$harvest.fish_amount_per_kg',
$harvest.breed_start_date',
.breed_end_date',
$harvest.total_weight_harvested',
{
$substr': [
'$toString': {
'$divide': [
{
'$subtract': [
'$harvest.breed_end_date',
'$harvest.breed_start_date',
],
},
],
24 * 60 * 60 * 1000, # Convert milliseconds to days
],
},
'$addFields': {
'fish_amount_per_kg': '',
'breed_start_date': '',
'breed_end_date': '$harvest.breed_end_date',
'total_weight_harvested': '',
'time_difference_in_days': '',
'$cond': {
'if': {
'$eq': [
{
},
0,
1,
],
},
'then': 1,
'else': {
},
}
},
```

```
[  
    '$subtract': [  
        '$harvest.breed_end_date',  
        '$harvest.breed_start_date',  
        * 60 * 1000, # Convert milliseconds to days  
    ],  
    '$divide': [  
        24 * 60  
    ],  
    '$project': {  
        "pond": 0,  
        "string": 0,  
        "suplemen": 0,  
        "feed": 0,  
        "fish": 0,  
        "pond": 0,  
    },  
],  
]  
harvest_aggregate = db.aggregate('harvest',  
harvest_pipeline)  
harvest_list = list(harvest_aggregate)  
harvest = harvest_list[0]  
print("harvest", harvest)  
  
##check if record already exist  
auction_draft_pipeline = [  
    {  
        "$match": {  
            "grading_id":  
ObjectID(harvest['grading_id']),  
        }  
    }  
]  
auction_draft_aggregate = db.aggregate(  
    auction_draft_collection_name, auction_draft_pipeline)  
auction_draft_list = list(auction_draft_aggregate)
```

```

## if len == 0, that mean the record is not exist, so
we create a new one
        if len(auction_draft_list) is 0:
            body = {
                "breeder_id": ObjectId(breeder_id),
                "harvest_id": ObjectId(harvest_id_form),
                "pond_activation_id": ObjectId(harvest['
pond_activation_id']),
                "grading_id": ObjectId(harvest['grading_id']),
                "fish_name": harvest['fish_name'],
                "fish_type": harvest['fish_type'],
                "fish_weight": harvest['total_weight_harvested
'],
                "fish_long": harvest['fish_long'],
                "fish_amount": harvest['fish_amount'],
                "recommended_price": harvest['price'],
                "breeder_price" : int(price),
                "pond_origin": harvest['pond_name'],
                "created_at": datetime.datetime.now(),
                "updated_at": datetime.datetime.now()
            }
            print(body)
            collection = db.get_collection(
auction_draft_collection_name)
            id = collection.insert_one(body)
            response = {
                "message": "success add data auction draft", "
id": id.inserted_id}
            response = json.dumps(response, default=str)
            return Response(response, mimetype="application/
json", status=200)
        else:
            response = {
                "message": "gagal, data auction draft sudah
dibuat"}
            response = json.dumps(response, default=str)
            return Response(response, mimetype="application/
json", status=400)
    except Exception as e:
        response = {"message": str(e)}
        response = json.dumps(response, default=str)
        return Response(response, mimetype="application/json",
status=400)

```

3.3 Fungsi untuk Mengambil Seluruh Data dari Tabel Auction Draft

```

auction_draft_collection_name = 'auction_draft'
class AuctionDraftsApi(Resource):

```

```

@jwt_required()
def get(self):
    try:
        current_user = get_jwt_identity()
        id = str(current_user['id'])
        breeder_id = ObjectId(id)
        pipeline = [
            {
                "$match": {
                    "breeder_id": breeder_id,
                }
            },
            {"$sort": {"created_at": 1}},
            {"$lookup": {
                "from": "harvest",
                "localField": "harvest_id",
                "foreignField": "_id",
                "as": "harvest"
            }},
            {"$unwind": {"path": "$harvest"}},
            {"$addFields": {
                "fish_harvested_id": "$harvest.
fish_harvested_id"
            }},
            {"$lookup": {
                "from": "fish_harvested",
                "localField": "pond_activation_id",
                "foreignField": "pond_activation_id",
                "as": "result"
            }},
            {"$unwind": {"path": "$result"}},
            {"$addFields": {
                "fish_amount_per_kg": "$result.
fish_amount_per_kg",
                "breed_start_date": "$result.
breed_start_date",
                "breed_end_date": "$result.
breed_end_date",
                "total_weight_harvested": "$result.
total_weight_harvested",
                "image": "$result.image",
                "fcr": "$result.fcr",
                "time_difference_in_days": {
                    "$divide": [
                        {"$subtract": ["$result.
breed_end_date", "$result.breed_start_date"]}], "24 * 60 * 60 * 1000
                }
            }},
            {"$project": {"result": 0, "harvest": 0}}
        ]
    
```

```

    {
      "$lookup": {
        "from": "auction",
        "localField": "_id",
        "foreignField": "auction_draft",
        "as": "result",
      },
    },
    {
      "$unwind": {
        "path": "$result",
        "preserveNullAndEmptyArrays": True,
      },
    },
    {
      "$match": {
        "result": {
          "$exists": False,
        },
      },
    },
  ]
}
auction_draft = db.aggregate(
  auction_draft_collection_name, pipeline)
list_auction_draft = list(auction_draft)
response = json.dumps(list_auction_draft, default=str)
return Response(response, mimetype="application/json",
status=200)
except Exception as e:
  response = {"message": str(e)}
  response = json.dumps(response, default=str)
  return Response(response, mimetype="application/json",
status=400)

```

3.4 Fungsi untuk Mengirimkan Data ke tabel Auction

```

auction_collection_name = 'auction'
class AuctionsApi(Resource):
  @jwt_required()
  def post(self):
    try:
      current_user = get_jwt_identity()
      breeder_id = str(current_user['id'])
      auction_draft_id_form = request.form.get("auction_draft",
                                              None)
      temp = auction_draft_id_form.split(',')
      final_auction_id = []
      for each in temp:
        final_auction_id.append(ObjectId(each))
    
```

```

        auction_name = request.form.get("auction_name", None)
        auction_start_date_form = request.form.get(
"auction\_start\_date", None)
            auction_end_date_form = request.form.get(
"auction\_end\_date", None)
                fish_type = request.form.get("fish_type", None)
                total_fish_amount = request.form.get("total_fish_amount",
", None)
                    total_fish_price = request.form.get("total_fish_price",
None)
                        total_fish_weight = request.form.get("total_fish_weight",
", None)

                date_format = '%Y-%m-%d %z'
                auction_start_date = datetime.datetime.strptime(
auction\_start\_date\_form, date_format)
                    auction_end_date = datetime.datetime.strptime(
auction\_end\_date\_form, date_format)

            body = {
                "breeder_id" : ObjectId(breeder_id),
                "auction_draft": final_auction_id,
                "auction_name": auction_name,
                "auction_start_date" : auction_start_date,
                "auction_end_date" : auction_end_date,
                "fish_type": fish_type,
                "total_fish_amount": int(total_fish_amount),
                "total_fish_price": int(total_fish_price),
                "total_fish_weight": float(total_fish_weight),
                "isAuctionFinish" : True,
                "created_at": datetime.datetime.now(),
                "updated_at": datetime.datetime.now()
            }
            print(body)
            collection = db.get_collection(auction_collection_name)
            id = collection.insert_one(body)
            response = {"message": "success add data auction", "id":
": id.inserted_id}
            response = json.dumps(response, default=str)
            return Response(response, mimetype="application/json",
status=200)
        except Exception as e:
            response = {"message": str(e)}
            response = json.dumps(response, default=str)
            return Response(response, mimetype="application/json",
status=400)

```

3.5 Fungsi untuk Mengambil Data dari Tabel Auction

```

auction_collection_name = 'auction'
class AuctionsApi(Resource):
    @jwt_required()
    def get(self):
        try:
            current_user = get_jwt_identity()
            breeder_id = str(current_user['id'])
            pipeline = [
                {
                    "$match": {
                        "breeder_id": {"$ne": ObjectId(breeder_id)},
                    },
                },
                {"$sort": {"created_at": -1}},
                {
                    "$lookup": {
                        "from": "breeder",
                        "localField": "breeder_id",
                        "foreignField": "_id",
                        "as": "breeder",
                    },
                },
                {
                    "$unwind": {
                        "path": "$breeder",
                        "preserveNullAndEmptyArrays": True,
                    },
                },
                {
                    "$addFields": {
                        "kabupaten_id": "$breeder.kabupaten_id",
                        "kecamatan_id": "$breeder.kecamatan_id",
                        "kelurahan_id": "$breeder.kelurahan_id",
                    },
                },
                {
                    "$project": {
                        "breeder": 0,
                    },
                },
            ]
            auction_draft = db.aggregate(auction_collection_name,
                pipeline)
            list_auction_draft = list(auction_draft)
            response = json.dumps(list_auction_draft, default=str)
            return Response(response, mimetype="application/json",
                status=200)
        
```

```

    except Exception as e:
        response = {"message": str(e)}
        response = json.dumps(response, default=str)
        return Response(response, mimetype="application/json",
status=400)

```

3.6 Fungsi untuk Mengirimkan Data ke Tabel Bid

```

collection_name = 'bid'
class BidsApi(Resource):
    @jwt_required()
    def post(self):
        try:
            current_user = get_jwt_identity()
            id = str(current_user['id'])
            breeder_id = ObjectId(id)
            breeder_name = str(current_user['name'])
            breeder_phone_number = str(current_user['phone'])
            auction_id = request.form.get("auction_id", None)
            breeder_bid = request.form.get("breeder_bid", None)

            body = {
                "auction_id": ObjectId(auction_id),
                "breeder_id": ObjectId(breeder_id),
                "breeder_name": breeder_name,
                "breeder_bid": int(breeder_bid),
                "breeder_phone_number": breeder_phone_number,
                "is_bid_accepted": False,
                "created_at": datetime.datetime.now(),
                "updated_at": datetime.datetime.now()
            }
            print(body)
            collection = db.get_collection(collection_name)
            id = collection.insert_one(body)
            response = {"message": "success add data bid", "id": id
            .inserted_id}
            response = json.dumps(response, default=str)
            return Response(response, mimetype="application/json",
status=200)
        except Exception as e:
            response = {"message": str(e)}
            response = json.dumps(response, default=str)
            return Response(response, mimetype="application/json",
status=400)

```

3.7 Fungsi untuk Mengambil Data dari Tabel Bid

```
collection_name = 'bid'
class BidsApiByAuctionId(Resource):
    @jwt_required()
    def get(self, auction_id):
        try:
            pipeline = [
                {
                    "$match": {
                        "auction_id": ObjectId(auction_id),
                    },
                },
                {
                    "$sort": {
                        "created_at": -1,
                    },
                },
                {
                    "$addFields": {
                        "breeder_bid": {
                            "$toInt": "$breeder_bid",
                        },
                    },
                },
                {
                    "$lookup": {
                        "from": "auction",
                        "localField": "auction_id",
                        "foreignField": "_id",
                        "as": "result",
                    },
                },
                {
                    "$unwind": {
                        "path": "$result",
                    },
                },
                {
                    "$addFields": {
                        "auction_end_date": "$result.
                    auction_end_date",
                    },
                },
                {
                    "$project": {
                        "result": 0,
                    },
                },
            ]
            print("pipeline", pipeline)
            auction_draft = db.aggregate(collection_name, pipeline)
            list_auction_draft = list(auction_draft)
```

```

        response = json.dumps(list_auction_draft, default=str)
        return Response(response, mimetype="application/json",
status=200)
    except Exception as e:
        response = {"message": str(e)}
        response = json.dumps(response, default=str)
        return Response(response, mimetype="application/json",
status=400)
    
```

3.8 Fungsi untuk Mengirim Data ke Tabel Transaction

```

collection_name = 'transaction'
class TransactionsApi(Resource):
    @jwt_required()
    def post(self):
        try:
            breeder_id = request.form.get("breeder_id",
None)
            bid_id = request.form.get("bid_id", None)
            final_bid_id = bid_id
            if bid_id != '':
                final_bid_id = ObjectId(bid_id)
            transaction_type = request.form.get("transaction_type", None)
            transaction_amount = request.form.get("transaction_amount", None)
            desc = request.form.get("desc", None)

            body = {
                "breeder_id": ObjectId(breeder_id),
                "bid_id": final_bid_id,
                "transaction_type": transaction_type,
                "transaction_amount": int(
transaction_amount),
                "desc": desc,
                "created_at": datetime.datetime.now(),
                "updated_at": datetime.datetime.now()
            }
            print(body)
            collection = db.get_collection(collection_name)
            id = collection.insert_one(body)
            response = {"message": "success add data
transaction", "id": id.inserted_id}
            response = json.dumps(response, default=str)
            return Response(response, mimetype="application
/json", status=200)
        except Exception as e:
            response = {"message": str(e)}
            response = json.dumps(response, default=str)
    
```

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

3.9 Fungsi untuk Mengambil Data dari Tabel Transaction

```
collection_name = 'transaction'
class TransactionsApi(Resource):
    @jwt_required()
    def get(self):
        try:
            current_user = get_jwt_identity()
            print("current_user", current_user)
            breeder_id = str(current_user['id'])
            pipeline = [
                {"$sort": {"created_at": -1}},
                {"$match": {"breeder_id": ObjectId(breeder_id)}}
            ]
            auction_draft = db.aggregate(collection_name, pipeline)
            list_auction_draft = list(auction_draft)
            response = json.dumps(list_auction_draft, default=str)
            return Response(response, mimetype="application/json", status=200)
        except Exception as e:
            response = {"message": str(e)}
            response = json.dumps(response, default=str)
            return Response(response, mimetype="application/json", status=400)
```

3.10 Fungsi untuk Mengirim Data ke Tabel Request Topup

```
class RequestTopupApi(Resource):
    @jwt_required()
    def post(self):
        try:
            current_user = get_jwt_identity()
            print("current_user", current_user)
            breeder_id = str(current_user['id'])
            topup_amount = request.form.get("topup_amount", None)
            file = request.files['image']
            if 'image' not in request.files:
                return {"message": "No file detected"}, 400
            if file.filename == '':
                return {"message": "No image selected for uploading
"}, 400
            if file and allowed_file(file.filename):
                filename = secure_filename(file.filename)
                path = os.path.join(current_app.instance_path,
current_app.config['UPLOAD_DIR'])
                image_path = os.path.join(path, filename)
```

```

        file.save(image_path)

    body = {
        "breeder_id": ObjectId(breeder_id),
        "topup_amount": int(topup_amount),
        "proof": filename,
        "is_accepted": False,
        "created_at": datetime.datetime.now(),
        "updated_at": datetime.datetime.now()
    }
    print(body)
    collection = db.get_collection('request_topup')
    id = collection.insert_one(body)
    response = {"message": "success add data transaction",
    "id": id.inserted_id}
    response = json.dumps(response, default=str)
    return Response(response, mimetype="application/json",
status=200)
except Exception as e:
    response = {"message": str(e)}
    response = json.dumps(response, default=str)
    return Response(response, mimetype="application/json",
status=400)

```

3.11 Fungsi untuk Mengambil Data dari Tabel Request Topup

```

class RequestTopupApi(Resource):
@jwt_required()
def get(self):
    try:
        current_user = get_jwt_identity()
        print("current_user", current_user)
        breeder_id = str(current_user['id'])
        pipeline = [
            {
                "$sort": {
                    "created_at": -1,
                },
                "$match": {
                    "is_accepted": False,
                },
            }
        ]
        aggre = db.aggregate('request_topup', pipeline)
        list_aggre = list(aggre)
    except:
        return {"message": "error"}, 500

```

```

        response = json.dumps(list_aggre, default=str)
        return Response(response, mimetype="application/json",
status=200)
    except Exception as e:
        response = {"message": str(e)}
        response = json.dumps(response, default=str)
        return Response(response, mimetype="application/json",
status=400)
    
```

3.12 Fungsi untuk Menyetujui Permintaan Topup

```

class RequestTopupApiById(Resource):
@jwt_required()
def put(self, id):
    try:
        body = {
            "is_accepted" : True,
            'updated_at' : datetime.datetime.now(),
        }
        print(body)
        collection = db.get_collection('request_topup')
        macthFilter = {
            "_id" : ObjectId(id)
        }
        updateFilter = {
            "$set" : body
        }
        collection.update_one(macthFilter, updateFilter)
        pipeline = [
            {
                "$sort": {
                    "created_at": -1,
                },
            },
            {
                "$match": {
                    "_id": ObjectId(id),
                },
            },
        ]
        print("pipeline", pipeline)
        aggre = db.aggregate('request_topup', pipeline)
        list_aggre = list(aggre)[0]
        transactionBody = {
            "breeder_id": ObjectId(list_aggre['breeder_id']),
            "bid_id": '',
            "transaction_type": 'IN',
        }
    
```

```

        "transaction_amount": int(list_aggre['topup_amount']
    ]),
    "desc": 'Topup',
    "created_at": datetime.datetime.now(),
    "updated_at": datetime.datetime.now()
}
collection = db.get_collection(collection_name)
idTransaction = collection.insert_one(transactionBody)
response = {
    "message": "success change data request topup", "id": id}
response = json.dumps(response, default=str)
return Response(response, mimetype="application/json",
status=200)
except Exception as e:
    response = {"message": str(e)}
    response = json.dumps(response, default=str)
    return Response(response, mimetype="application/json",
status=400)

```

3.13 Fungsi untuk Mengambil Data Jumlah Saldo

```

class SaldoApi(Resource):
@jwt_required()
def get(self):
    try:
        current_user = get_jwt_identity()
        print("current_user", current_user)
        breeder_id = str(current_user['id'])
        ## get total saldo masuk
        pipeline_in = [
            {
                "$match": {
                    "breeder_id": ObjectId(breeder_id),
                    "transaction_type": "IN",
                }
            },
            {
                "$group": {
                    "_id": {
                        "breeder_id": "$breeder_id",
                    },
                    "total_amount": {
                        "$sum": {
                            "$toInt": "$transaction_amount",
                        }
                    },
                }
            }
        ]

```

```
        },
        {
            "$addFields": {
                "breeder_id": "$_id.breeder_id",
            },
        },
        {
            "$project": {
                "_id": 0,
            },
        },
    ],
    saldo_in_aggregate = db.aggregate(collection_name,
pipeline_in)
    list_saldo_in_aggregate = list(saldo_in_aggregate)
    saldo_in = 0
    if len(list_saldo_in_aggregate) is not 0:
        saldo_in = list_saldo_in_aggregate[0]['total_amount']
    ]

    # get total saldo keluar
    pipeline_out = [
        {
            "$match": {
                "breeder_id": ObjectId(breeder_id),
                "transaction_type": "OUT",
            },
        },
        {
            "$group": {
                "_id": {
                    "breeder_id": "$breeder_id",
                },
                "total_amount": {
                    "$sum": {
                        "$toInt": "$transaction_amount",
                    },
                },
            },
        },
        {
            "$addFields": {
                "breeder_id": "$_id.breeder_id",
            },
        },
        {
            "$project": {
                "_id": 0,
            },
        },
    ]
```

```

        saldo_out_aggregate = db.aggregate(collection_name,
pipeline_out)
        list_saldo_out_aggregate = list(saldo_out_aggregate)
saldo_out = 0
if len(list_saldo_out_aggregate) is not 0:
    saldo_out = list_saldo_out_aggregate[0]['
total_amount']

saldo = 0
if saldo_in and saldo_out is not 0:
    saldo = int(saldo_in - saldo_out)
elif saldo_in is not 0:
    saldo = saldo_in
elif saldo_out is not 0:
    saldo = saldo_out

response = {"saldo" : saldo}
response = json.dumps(response, default=str)
return Response(response, mimetype="application/json"
, status=200)
except Exception as e:
    response = {"message": str(e)}
    response = json.dumps(response, default=str)
    return Response(response, mimetype="application/json"
, status=400)

```

3.14 Implementasi Route Server

```

from .controller.harvest import *
from .controller.auction_draft import *
from .controller.auction import *
from .controller.bid import *
from .controller.transaction import *
def initialize_routes(api):
    ## get, edit, delete harvest by pond activation id for auction
    draft
    api.add_resource(HarvestApi, '/api/harvests/<pond_activation_id
>')

    ## get and post auction draft
    api.add_resource(AuctionDraftsApi, '/api/auction/drafts')

    ## get, update, and delete auction draft by id
    api.add_resource(AuctionDraftApi, '/api/auction/draft/<id>')

    ## get and post auction
    api.add_resource(AuctionsApi, '/api/auctions')

```

```
## get, update and delete auction by id
api.add_resource(AuctionApi, '/api/auction/<id>')

## get auction by breeder id
api.add_resource(AuctionApiByBreederId, '/api/auctions/<breeder_id>')

## get and post bid
api.add_resource(BidsApi, '/api/bids')
## get bid by breeder id to get breeder's bid history
api.add_resource(BidsApiByBreederId, '/api/bids/<breeder_id>')
## get bid by auction id to get bid list in an auction
api.add_resource(BidsApiByAuctionId, '/api/bids/auction/<auction_id>')

## get and post transactions
api.add_resource(TransactionsApi, '/api/transactions')
api.add_resource(RequestTopupApi, '/api/request/topup')
api.add_resource(RequestTopupApiById, '/api/request/topup/<id>')

## get and delete transaction by id
api.add_resource(TransactionApi, '/api/transaction/<id>')

## get saldo
api.add_resource(SaldoApi, '/api/saldo')
```

*Mencerdaskan dan
Memartabatkan Bangsa*

LAMPIRAN IV

Transkrip Wawancara Pak Umar

4.1 Fitur Rekomendasi Harga

Pak Eka: Untuk selanjutnya, kami membutuhkan konfirmasi dari Bapak terkait fitur rekomendasi harga, Pak. Fahreza, mohon Anda yang menjelaskan.

Fahreza: Baik Pak, komponen-komponen yang dihitung untuk menjadi rekomendasi harga meliputi benih, pakan, suplemen, listrik, dan infrastruktur yang digunakan.

Pak Umar: Aset ini dimaksudkan sebagai biaya pengurangan, bukan?

Pak Eka: Bukan, Pak. Ini mencakup infrastruktur.

Pak Umar: Iya, misalnya biaya pembuatan kolam 5 juta yang bertahan selama 5 tahun, berarti per tahunnya dihitung 1 juta?

Pak Eka: Iya, Pak. Kami memang sudah membaginya selama 5 tahun secara otomatis agar harga ikan tidak terlalu tinggi.

Pak Umar: Bagaimana jika kita bagi per siklus panen untuk aset itu?

Pak Eka: Benar, Pak. Aset itu dibagi dengan jumlah kolam aktif. Misalnya, ada 10 kolam aktif dengan biaya pembuatannya masing-masing mencapai 5 juta, berarti total cost-nya 50 juta. Kemudian, apabila pembudidaya ingin menjual ikan, 50 juta tersebut dibagi dengan siklus musim sekarang, jumlah kolam aktif, dan 5 tahun, seperti yang sudah kami set untuk aset tersebut kembali dalam waktu 5 tahun, tidak lebih cepat maupun lebih lambat. Namun, kami dapat mempertimbangkan perubahan menjadi 3 tahun atau lebih lama dari 5 tahun, sesuai dengan masukan dari Bapak.

Pak Umar: Jadi, untuk aset ini, masuknya sebagai biaya penyusutan, ya? Dengan harapan setelah 5 tahun, kita (pembudidaya) ingin membuat kolam baru dan dana sudah tersedia, begitu kan?

Pak Eka: Iya, Pak. Sama seperti listrik, juga kami bagi dengan total jumlah kolam aktif. Silakan lanjutkan, Fahreza.

Fahreza: Baik, Pak. Seluruh komponen harga tersebut dijumlahkan, kemudian dibagi dengan jumlah ikan hidup.

Pak Umar: Ya, jumlah hasil panen.

Pak Eka: Namun, yang membuat perbedaan, untuk saat ini masih dibagi berdasarkan jumlah ekor ikan hidup, Pak. Jadi, harganya untuk per ekor.

Pak Umar: Untuk kenyamanan, harganya sebaiknya dibagi per kilogram, karena kami biasanya menjual 1 kg isi 10, 1 kg isi 5 untuk asap, dan 1 kg isi 78 untuk restoran. Jadi, lebih realistik jika dibagi per kilogram. Sebenarnya, untuk fitur harga ini, meskipun belum melakukan panen, namun berdasarkan pengalaman, harga per kilogramnya sudah dapat diprediksi, mulai dari benih, pakan, vitamin, dan lain-lain.

Pak Eka: Namun, masih ada fluktuasi, bukan?

Pak Umar: Iya, benar. Jadi, per siklus panen akan menghasilkan perbedaan.

Pak Eka: Jadi, kami melaporkannya ke dinas, nanti dari dinas akan melaporkan ke kkp. Mereka dapat melihat pergerakan harga berdasarkan data dari kami, Pak. Gunanya aplikasi kami, ketika pembudidaya membeli pakan, jumlah pembeliannya akan tercatat. Jadi, jika ada kenaikan harga pakan, dan dari data di aplikasi kami ternyata pembelian pelet sedikit, harapannya dapat menahan fluktuasi harga tersebut, Pak.

Pak Umar: Baik, sangat bagus, Pak.

4.2 Fitur Lelang

Azzam: Selamat sore, Pak Umar. Saya Azzam dari tim riset Aqua Breeding Ilmu Komputer UNJ. Kami sangat menghargai kesempatan ini untuk mendapatkan masukan dari Bapak, seorang pembudidaya besar di Parung, terkait dengan pengembangan aplikasi Aqua Breeding versi ketiga kami. Kami berencana untuk menambahkan fitur lelang, dan kami ingin mendengar pandangan serta masukan dari Bapak agar aplikasi ini dapat lebih bermanfaat dan sesuai dengan kebutuhan di lapangan.

Pak Umar: Selamat sore, Azzam. Saya senang bisa berkontribusi pada pengembangan aplikasi Anda. Silakan lanjutkan.

Azzam: Jadi, untuk aplikasi Aquabreeding versi ketiga akan ada fitur sistem lelang. Apabila pembudidaya telah melakukan panen, maka akan ditampilkan data ikan panen tersebut. Bapak dapat melelang ikan tersebut ke pembudidaya lain. Ketika Bapak mengklik opsi lelang, Bapak akan diarahkan ke halaman draft auction lelang. Di halaman ini, akan ada harga penawaran awal yang merupakan rekomendasi harga dari cost pembudidaya selama satu musim budidaya dari awal hingga panen—

Pak Umar: Ya, sebagai harga dasar.

Azzam: Benar, Pak. Ini bersifat sebagai rekomendasi harga saja, di mana pembudidaya dapat menambahkan keuntungan ke dalam harga tersebut. Sebagai contoh, jika harga penawaran awalnya 26rb dan Bapak ingin mengambil untung 10rb, maka harga akhirnya dapat ditambah hingga 36rb. Setelah data kg panen dan harga penawaran awal sesuai, Bapak akan diarahkan ke halaman draft lelang ini. Bapak dapat melakukan lelang beberapa jenis ikan sekaligus.

Pak Umar: Oh, ya. Saya paham.

Azzam: Selanjutnya, Bapak hanya perlu mengklik tombol "Pilih ini," dan kemudian Bapak akan diarahkan ke halaman konfirmasi lelang. Di sini, Bapak akan diminta untuk memastikan data lelang ikan sudah benar. Selanjutnya, Bapak akan diminta untuk mengatur tanggal lelang dimulai dan tanggal lelang berakhir, misalnya, durasi lelang selama seminggu, dua minggu, atau sebulan.

Pak Umar: Ooooh, ya. Saya paham.

Azzam: Setelah selesai, Bapak akan diarahkan ke halaman lelang saya. Ketika Bapak mengklik tombol "Lihat Detail," Bapak akan diarahkan ke halaman detail lelang. Di halaman ini, akan terdapat informasi detail lelang, jenis ikan yang

dilelang, dan daftar penawaran dari pembudidaya lain. Jika ada pembudidaya lain yang tertarik untuk menawar ikan Bapak, penawaran tersebut akan muncul di halaman ini. Bapak dapat melihat siapa saja yang mengajukan penawaran dan jumlah penawaran mereka. Jika Bapak tertarik dengan salah satu penawaran, Bapak dapat menerima penawaran tersebut dengan mengklik tombol "Terima Penawaran" dan melakukan konfirmasi. Tampilan halaman tersebut akan berubah untuk menampilkan penawaran terpilih dan uangnya akan langsung masuk ke akun Bapak. Bapak juga dapat berkoordinasi lebih lanjut dengan penawar terpilih melalui nomor telepon yang tertera.

Pak Umar: Jadi, lelang ini antar pembudidaya?

Pak Eka: Ya, benar Pak. Inilah sebabnya disebut sebagai sistem interkoneksi pembudidaya. Namun, setelah berdiskusi dengan dinas, kami memperluasnya untuk mencakup distributor juga.

Pak Umar: Bagi pembudidaya masih bisa, karena mereka dapat berkomunikasi dengan kelompoknya. Misalnya, ada kelompok A, B, dan C, dan masing-masing kelompok memiliki satu perwakilan yang memiliki pasokan atau kebutuhan pasar. Jika kelompok A memiliki kebutuhan pasar, mereka dapat mengajukan penawaran kepada kelompok B, yang mungkin melelang hasil panennya. Artinya, seperti itu, bukan?

Azzam: Iya, benar Pak.

Pak Eka: Tetapi, yang membedakannya, pembudidaya tidak dapat sembarangan menentukan harga. Harga akan direkomendasikan berdasarkan biaya yang dikeluarkan selama satu musim budidaya.

Pak Umar: Ya, sebagai harga pokok.

Pak Eka: Benar. Dengan sistem lelang ini, pembudidaya dapat menentukan margin keuntungan.

Pak Umar: Baik. Mengenai durasi lelang tadi, untuk beberapa jenis ikan seperti ikan lele, mungkin perlu dibatasi waktu lelangnya menjadi 3 hari saja. Karena, katakanlah, lele ukuran 6-7, dalam waktu 3 hari dapat tumbuh menjadi 7-8. Jadi, untuk beberapa jenis ikan tertentu, batas waktu lelang dapat disesuaikan, bukan?

Azzam: Tentu, Pak.

Pak Umar: Hal ini juga dapat mempengaruhi faktor pakan. Pastikan transaksi disepakati agar kita sebagai pembudidaya tidak mengalami kerugian karena biaya pakan selama 3 hari. Harga yang dihitung berdasarkan cost selama satu musim budidaya harus mencakup periode menunggu transaksi disetujui.

Azzam: Saya paham, Pak.

Pak Umar: Untuk beberapa jenis ikan tertentu, perlu dibatasi. Sebagai contoh, transaksi harus selesai dalam waktu 3 hari. Sebab, jika ikan yang dilelang berjumlah 10rb ekor, dan setiap ekor membutuhkan pakan sebanyak 5

Pak Eka: Dengan begitu, untuk memfasilitasi hal tersebut, kita dapat menambahkan fitur baru.

Pak Umar: Nah, ada formula yang dapat diterapkan di sini. Misalnya, jika transaksi berlangsung selama 3 hari, maka total biaya tambahan sebesar 1jt 800rb

tidak dimasukkan ke dalam harga penawaran awal.

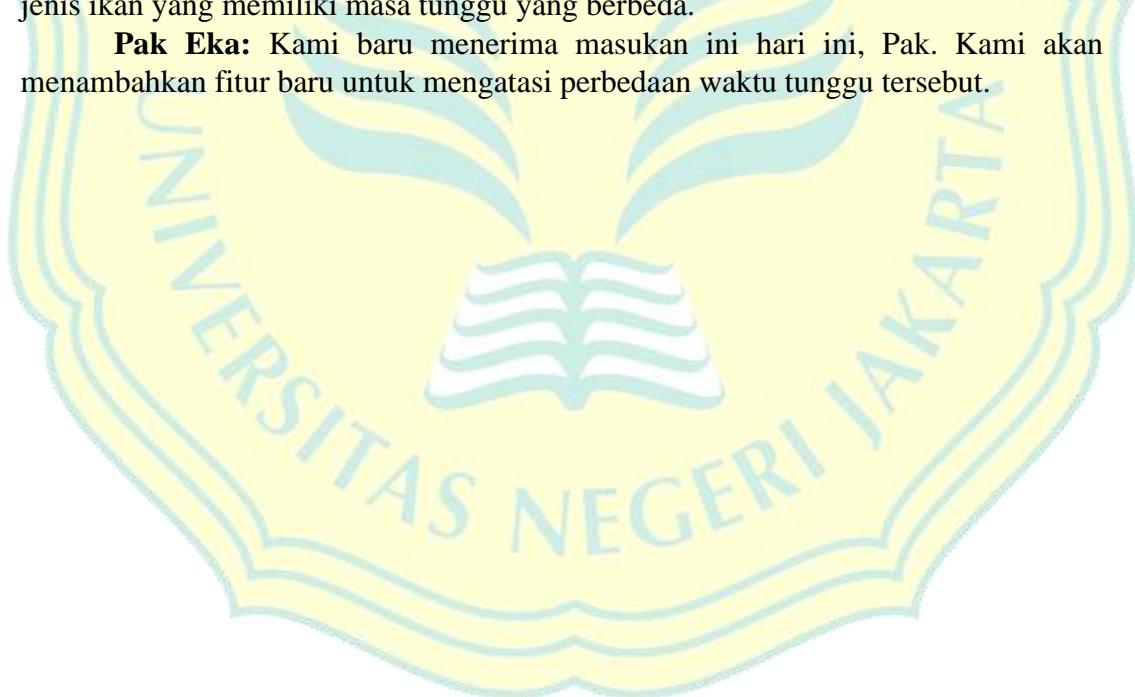
Pak Eka: Jadi, berarti perlu ada tambahan fitur waktu tunggu yang akan mempengaruhi harga. Bagaimana jika biaya waktu tunggu ini dibagi antara pembudidaya yang melelang dan pembudidaya yang mengajukan penawaran?

Pak Umar: Benar, itu dapat dijalankan. Untuk mengelola risiko biaya pakan selama waktu tunggu, harga penawaran awal dapat ditambah dengan biaya yang mungkin timbul selama periode tersebut. Selanjutnya, perlu ada langkah-langkah konkret untuk memastikan komitmen transaksi.

Pak Eka: Hal itu telah kami tangani, Pak. Kami menggunakan sistem tabungan bersama, di mana setiap pihak harus menyetor uang ke pengelola sebelum melakukan transaksi. Jadi, jika pembudidaya tidak melakukan penyetoran, mereka tidak dapat melakukan transaksi.

Pak Umar: Bagus, itu memberikan jaminan yang baik. Pastikan bahwa setiap transaksi memiliki keamanan tersendiri. Namun, kembali ke masukan saya tadi, ada jenis ikan yang memiliki masa tunggu yang berbeda.

Pak Eka: Kami baru menerima masukan ini hari ini, Pak. Kami akan menambahkan fitur baru untuk mengatasi perbedaan waktu tunggu tersebut.



*Mencerdaskan dan
Memartabatkan Bangsa*

LAMPIRAN V

Transkrip Wawancara dengan Pak Ending

5.1 Fitur Penentuan Harga dan Sistem Lelang

Konteks: Pak Ending adalah seorang petani yang sedang belajar budidaya ikan. Kami memutuskan untuk mewawancarai beliau sebagai orang awam di bidang budidaya perikanan.

Pak Eka: Jadi, aplikasi yang kami kembangkan adalah aplikasi budidaya ikan. Kami ingin membuat aplikasi yang melakukan pelacakan pengeluaran yang dilakukan oleh pembudidaya ikan.

Pak Ending: Kalau soal biaya ikan saya belum terlalu paham, Pak, karena ini sebatas hobi. Saya memberikan jenis pakan apa saja yang penting ikan tumbuh besar.

Pak Eka: Prinsipnya sama dengan bertani, Pak. Anggap saja ada aplikasi yang mencatat pengeluaran pupuk, berapa habisnya. Kemudian, harga diatur tidak jauh dari itu sehingga harga pasti di atas. Apakah Bapak setuju dengan pendekatan tersebut?

Pak Ending: Setuju saja saya, Pak.

Pak Eka: Tinggal masalahnya begini, Pak. Apabila ada pengeluaran, seperti membeli ikan, apakah petani mau melakukan pembukuan?

Pak Ending: Kalau memang diarahkan pasti mau, Pak.

Pak Eka: Berarti ada potensi menuju sana, bukan, Pak?

Pak Ending: Iya, Pak.

Pak Eka: Lalu, Pak, pada hari Senin lalu ketika kami mempresentasikan mengenai transaksi sistem lelang yang akan ditambahkan ke aplikasi Aquabreeding, terkait dengan transaksi tersebut, apakah Bapak setuju bahwa transaksi tersebut dapat mempercepat alur transaksi jika kami menambahkan fitur lelang?

Pak Ending: Ikan yang bisa dijual jenis apa saja?

Pak Eka: Bisa jenis apapun, seperti ikan lele, ikan mas, ikan gurame, ikan nila, semuanya bisa ditransaksikan di dalam aplikasi. Model penjualan petani biasanya tidak langsung ke eceran, mayoritas dijual ke distributor. Oleh karena itu, kami mengembangkan fitur transaksi lelang ini agar pembudidaya bisa langsung menjual ke eceran tanpa terpaku pada distributor lagi. Apakah menurut Bapak fitur tersebut memiliki potensi untuk meningkatkan kesejahteraan petani dan dapat dilakukan dengan lancar?

Pak Ending: Kalau dari pengalaman saya sebagai petani cabai, jika hasil panen sedikit, saya tidak berani menjual ke pasar. Saya langsung menjual ke restoran, sekitar 2 kg atau 3 kg.

Pak Eka: Karena selisih harganya lebih tinggi, bukan? Karena jika dijual ke distributor, mereka akan memotong lagi, bukan?

Pak Ending: Iya, benar.

Pak Eka: Jika begitu, berarti ada satu masalah jika Bapak langsung menjual

ke eceran, yaitu produksinya tidak bisa banyak karena mereka tidak bisa menampung. Sama seperti di budidaya ikan, jika saya menargetkan harga tinggi, produksinya tidak banyak. Saya menjual ke konsumen tertentu yang bersedia membayar harga tinggi. Namun, mayoritas jika produksi ikannya banyak, saya menjual ke tengkulak, meskipun margin keuntungannya kecil. Oleh karena itu, kami mengembangkan transaksi lelang di aplikasi Aqua Breeding kami untuk mengatasi masalah ini, Pak.

Pak Ending: Iya, bagus, Pak.

5.2 Validasi Fitur Rekomendasi Harga

Fahreza: Jadi, ini adalah aplikasi yang telah kami rancang, Pak. Aplikasi ini berfungsi untuk mencatat informasi seperti masa budidaya, pemberian pakan, jumlah panen, kondisi air, dan data lainnya. Dari aplikasi ini, apakah Bapak bersedia berbagi data budidaya ikan dengan orang lain?

Pak Ending: Kalau soal kondisi air, saya tidak punya informasi detail, untuk saya, asalkan ada air, saya langsung masukkan ikan, itu sudah cukup.

Fahreza: Jadi, jika kita hanya mencatat pemberian pakan, misalnya jumlah total pakan yang Bapak berikan, maka kita bisa menghitung total berapa kilogram pakan yang digunakan dan juga menghitung modal yang Bapak keluarkan. Dengan begitu, kita bisa menentukan harga minimal jual ikan. Bagaimana menurut Bapak? Apakah bersedia menggunakan aplikasi ini untuk mencatat data selama musim budidaya ikan?

Pak Ending: Tentu, saya setuju, Mas!

Fahreza: Setelah mengisi data pakan tadi, bagaimana menurut Bapak? Apakah sudah memenuhi kebutuhan Bapak sebagai pembudidaya ikan?

Pak Ending: Sudah cukup bagus, Mas.

*Mencerdaskan dan
Memartabatkan Bangsa*

LAMPIRAN VI

Transkrip Wawancara dengan Pak Sobari

Fikri: Baik Pak, jadi hari ini kami hendak mendemonstrasikan aplikasi Aquabreeding kepada Bapak, dan kami butuh pendapat serta validasi dari Bapak bahwa aplikasi Aquabreeding kami telah memenuhi kebutuhan Bapak sebagai pembudidaya ikan Pak. **Pak Sobari:** Baik mas, silahkan dimulai. **Fikri:** Baik Pak, pertama-tama mohon unduh terlebih dahulu Pak aplikasinya, sudah saya kirimkan ke WhatsApp Bapak. **Pak Sobari:** Sudah mas **Fikri:** Kalau sudah, Bapak bisa buka aplikasi nya Pak, ketika pertama kali dibuka Bapak akan diarahkan ke halaman login Pak. Namun, karena Bapak belum punya akun, silahkan klik Register terlebih dahulu Pak. **Pak Sobari:** Yang ini mas? **Fikri:** Benar Bapak, kemudian mohon isikan formulir pendaftarannya Pak **Pak Sobari:** KTP... ini 16 digit ya? **Pak Eka:** Benar Pak, namun untuk keperluan demo, tidak perlu diisikan dengan nomor KTP asli Bapak **Pak Sobari:** Baik... 1 nya 4 kali... 2 nya 4 kali... 3 nya 4 kali... dan 4 nya 4 kali... sudah mas **Fikri:** Kemudian, mohon isikan breederId nya Pak, untuk digunakan ketika login, bisa diisikan menggunakan nama Bapak saja Pak **Pak Sobari:** Baik mas, sudah mas **Fikri:** Baik Pak, bila sudah mohon isikan password nya Pak, untuk keperluan login juga Pak **Pak Sobari:** Passwordnya? apa ya mas? **Pak Eka:** Samakan saja dengan nama Bapak **Fikri:** Silahkan masukkan nama farm Bapak Pak **Pak Sobari:** Namanya apa ya... Cimahok saja ya **Fikri:** Ini alamat bisa diisi, samakan saja Cimahok Pak. Apabila sudah, bisa langsung register saja Pak. Untuk menambahkan kolam baru, Bapak bisa ke halaman Daftar Kolam Pak, lalu isikan nama kolamnya Pak **Pak Sobari:** Baik, saya isikan nama kolamnya Kolam Utama saja ya **Fikri:** Boleh Pak, untuk ukurannya persegi ya Pak? **Pak Sobari:** Iya mas, ukurannya 12×10 mas **Fikri:** Untuk ketinggian airnya bisa diisi Pak **Pak Sobari:** 1,5 meter mas **Fikri:** Silahkan klik tombol Registrasi Kolam **Pak Eka:** Langsung aktivasi saja **Fikri:** Baik Pak, untuk memulai musim budidaya, Bapak bisa klik kolam yang baru saja Bapak buat, kemudian Bapak bisa memilih jenis ikan budidayanya, apakah pembesaran atau benih? **Pak Sobari:** Oh ini ikan kelas pembesaran mas **Fikri:** Baik Pak, silahkan pilih kelas pembesaran Pak, klik Nila Merah dan Ikan Mas, untuk bobotnya estimasi saja Pak, sekitar 84 ekor, beratnya 14 kg. Tinggi air ini kira-kira berapa ya Pak? **Pak Sobari:** 1 meter mas **Fikri:** Baik Pak, bila sudah silahkan klik tombol Aktivasi Pak. Setelah ini kita coba untuk beri pakan Pak. Untuk hari ini, Bapak sudah isi berapa kilogram Pak? **Pak Sobari:** Sekitar 2,5 kilo mas **Fikri:** Baik, silahkan klik jenis pakannya pelet Pak, kemudian isikan jumlah pemberian pakannya sebesar 2,5 kilogram Pak. **Pak Sobari:** Sudah mas **Fikri:** Bila sudah, silahkan klik tombolnya Pak. Maka data pemberian pakan Bapak sudah tercatat dalam aplikasi Pak. Setelah ini kita akan coba Panen Pak. Untuk Panen, Bapak silahkan klik back Pak, kemudian klik tombol Panen ini Pak. Untuk kasus Bapak, berarti panen ini diisikan setiap ada pancing Pak, silahkan isi data bobot panen nya Pak, nanti akan bisa terdeteksi Pak jumlah ikan dan bobot ikan

yang sudah Bapak panen Pak. **Pak Sobari:** Baik mas, paham paham **Fikri:** Bagaimana pendapat Bapak setelah menggunakan aplikasi ini Pak? Apakah sudah menjawab kebutuhan Bapak? **Pak Sobari:** Oh iya sudah mas **Fikri:** Apakah Bapak bisa memberikan validasi terhadap aplikasi ini Pak? **Pak Sobari:** Bisa mas, aplikasi ini saya validasi sudah menjawab kebutuhan saya sebagai pembudidaya ikan mas.



*Mencerdaskan dan
Memartabatkan Bangsa*

LAMPIRAN VII

Kesimpulan Wawancara dengan Pak Umar terkait fitur Lelang

Pada tanggal 29 Desember 2023, diadakan rapat dengan Pak Umar selaku Ketua Kelompok Pintu Air, yakni kelompok pembudidaya ikan air tawar yang berlokasi di Desa Jampang, Kecamatan Kemang, Kabupaten Bogor yang membahas fitur sistem lelang, sistem multiuser dan sistem keuangan. Setelah mendengarkan pemaparan aplikasi dari peneliti, beliau menyetujui bahwa fitur-fitur tersebut berguna bagi pembudidaya dan setuju untuk bekerja sama dalam menguji aplikasi versi ketiga yang rencananya akan diadakan pada bulan Februari 2024. Beliau memberikan saran pada harga penawaran lelang, yakni harga tersebut harus bertambah seiring berjalannya waktu hingga ikan yang dilelang berhasil terjual. Misal, harga penawaran awal ikan lele sebesar Rp. 100.000 per kg, harga penawaran awal tersebut akan bertambah Rp. 500/hari, sehingga apabila ikan lele tersebut laku pada hari keempat setelah lelang dimulai, maka harga akhir dari lelang tersebut menjadi sebesar Rp. 102.000/kg.



Gambar 7.1: Pertemuan dengan Pak Umar

*Mencerdaskan dan
Memartabatkan Bangsa*

DAFTAR RIWAYAT HIDUP



Nama : Abdullah Azzam
Tempat, Tanggal Lahir : Bekasi, 13 September 2000
Email : abdullah.azzam130@gmail.com

Pendidikan

1. SD Negeri Babelan Kota 01 (2006-2012)
2. SMP Negeri 01 Babelan (2012-2015)
3. SMA Negeri 01 Babelan, IPA (2015-2018)
4. Institut Negeri Sumatera, Fisika (2018-2019)
5. Universitas Negeri Jakarta, Ilmu Komputer (2019-2024)

Pengalaman Organisasi

1. Sekretaris, DEFAULT (April 2021 - Januari 2022)

Prestasi

1. Mendapatkan sertifikasi Android Associate Developer pada tahun 2022

Pengalaman Kerja

1. Android Developer Intern, **PT Fhadira Inovasi Teknologi** (Agustus - Desember, 2022)
2. Android Developer, **PT Fhadira Inovasi Teknologi** (Januari - Sekarang)

METADATA

Judul : Aplikasi Aquabreeding dengan Integrasi Fitur Multiuser Pembudidaya, Sistem Lelang dan Sistem Keuangan sebagai Sistem Interkoneksi Petani
Nama : Abdullah Azzam
NIM : 1313619028
Pembimbing I : Muhammad Eka Suryana, M.Kom
Pembimbing II : Med Irzal, M.Kom
Keyword : 1. Lelang
 2. Panen
 3. Transaksi
 4. Budidaya Ikan



*Mencerdaskan dan
Memartabatkan Bangsa*