

**PENGEMBANGAN SISTEM MONITORING MESIN DIESEL BERBASIS
IOT PADA PT BISMA JAYA MENGGUNAKAN METODE EXTREME
PROGRAMMING (XP)**



Oleh
Haidar Dzaky Sumpena
10201043

Pembimbing Utama : Aidil Saputra Kirsan, S.Kom., M.Tr.Kom
Pembimbing : Henokh Lugo Hariyanto, M.Sc.
Pendamping

**PROGRAM STUDI SISTEM INFORMASI
JURUSAN MATEMATIKA DAN TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI KALIMANTAN
BALIKPAPAN**

2024

Halaman ini sengaja dikosongkan

LEMBAR PERSETUJUAN

Proposal Tugas Akhir dengan judul :

**”PENGEMBANGAN SISTEM MONITORING MESIN DIESEL BERBASIS
IOT PADA PT BISMA JAYA MENGGUNAKAN METODE EXTREME
PROGRAMMING (XP)”**

Yang disusun oleh:

Haidar Dzaky Sumpena

NIM. 10201043

Telah diperiksa dan disetujui oleh dosen pembimbing:

Pembimbing Utama

Pembimbing Pendamping

Aidil Saputra Kirsan, S.Kom., M.Tr.Kom
NIPH. 100320240

Henokh Lugo Hariyanto, M.Sc.
NIP. 199303062022041001

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur kepada Allah SWT atas berkat dan rahmat-Nya penulis dapat menyelesaikan laporan tugas akhir yang berjudul:

**”PENGEMBANGAN SISTEM MONITORING MESIN DIESEL BERBASIS IOT
PADA PT BISMA JAYA MENGGUNAKAN METODE EXTREME
PROGRAMMING (XP)”**

Laporan tugas akhir ini merupakan salah satu syarat yang harus ditempuh untuk menyelesaikan Program Sarjana di Program Studi Sistem Informasi, Jurusan Matematika dan Teknologi Informasi, Institut Teknologi Kalimantan (ITK) Balikpapan. Besar terima kasih penulis ucapkan kepada:

1. Bapak Aidil Kirsan Saputra, S.Kom., M.Tr.Kom, selaku Dosen Pembimbing Utama dan Bapak Henokh Lugo Hariyanto, S.Si., M.Sc., selaku Dosen Pembimbing Pendamping.
2. Ibu Sri Rahayu Natasia, S.Komp, M.Si., M.Sc., selaku Koordinator Program Studi Sistem Informasi Jurusan Matematika dan Teknologi Informasi ITK.
3. Bapak/Ibu Dosen dan Bapak/Ibu Tendik Program Studi Sistem Informasi Jurusan Matematika dan Teknologi Informasi ITK.
4. Serta semua pihak yang terlibat dalam penyusunan proposal tugas akhir ini.

Penulis menyadari bahwa laporan tugas akhir ini masih jauh dari sempurna dikarenakan terbatasnya pengalaman dan pengetahuan yang dimiliki penulis. Oleh karena itu, penulis mengharapkan segala bentuk saran serta masukan bahkan kritik yang membangun dari berbagai pihak. Semoga tulisan ini memberikan manfaat kepada semua pihak yang membutuhkan dan terutama untuk penulis.

Balikpapan, 16 Januari 2024

Haidar Dzaky Sumpena
NIM 10201043

Halaman ini sengaja dikosongkan

PENGEMBANGAN SISTEM MONITORING MESIN DIESEL BERBASIS IOT PADA PT BISMA JAYA MENGGUNAKAN METODE EXTREME PROGRAMMING (XP)

Nama : Haidar Dzaky Sumpena
NIM : 10201043
Dosen Pembimbing Utama : Aidil Kirsan Saputra, S.Kom., M.Tr.Kom
Dosen Pembimbing Pendamping : Henokh Lugo Hariyanto, M.Sc.

ABSTRAK

Dalam penelitian ini telah dimanfaatkan IoT dalam pemantauan jumlah bahan bakar di industri maritim yang dipasang pada kapal milik PT Bisma Jaya. Pembangunan sistem IoT ini didasarkan oleh metode *Extreme Programming* yang menekankan kolaborasi serta pengembangan dinamis. Dipilihnya metode ini juga terbukti mampu menangani umpan balik dan menjaga kualitas kode selama pengembangan sistem berlangsung. Luaran dari penelitian ini adalah Sistem Monitoring berbasis web yang menggunakan teknologi Django dan NextJS. Hasilnya, telah dibangun sistem pemantauan yang dipasang pada 3 kapal dengan interval pengambilan data 1 menit secara konstan. Pengembangan lebih lanjut akan difokuskan pada keselamatan kapal dan pemantauan emisi.

Kata Kunci : *Internet of Things*, sistem *monitoring*, bahan bakar, *Extreme Programming*

Halaman ini sengaja dikosongkan

DAFTAR ISI

LEMBAR PERSETUJUAN	i
KATA PENGANTAR	ii
ABSTRAK	iii
DAFTAR ISI	iv
DAFTAR TABEL	vii
DAFTAR GAMBAR	viii
DAFTAR LAMPIRAN	x
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	4
1.3 Tujuan	4
1.4 Manfaat	4
1.5 Batasan Penelitian	5
1.6 Kerangka Pemikiran Penelitian	5
BAB II TINJAUAN PUSTAKA	8
2.1 PT Bisma Jaya	8
2.2 <i>Internet of Things</i>	9
2.3 Raspberry Pi	10
2.4 Perbandingan SDLC	11
2.5 <i>Extreme Programming (XP)</i>	16
2.6 <i>User Story</i>	18
2.7 <i>Entity Relationship Diagram</i>	18
2.8 MySQL	20
2.9 <i>Application Programming Interface</i>	20

2.10	Django	21
2.11	NextJS	21
2.12	<i>Whitebox Testing</i>	22
2.13	<i>Blackbox Testing</i>	24
2.14	Metode Perhitungan Bahan Bakar	25
2.15	Penelitian Terdahulu	27
	BAB III METODE PENELITIAN	32
3.1	Garis Besar Penelitian	32
3.2	Diagram Alir Penelitian	32
3.3	Prosedur Penelitian	32
3.3.1	Identifikasi Masalah	33
3.3.2	Studi Literatur	33
3.3.3	Pengembangan Sistem	34
	BAB IV HASIL DAN PEMBAHASAN	39
4.1	Perancangan (<i>Exploration Phase</i>)	39
4.1.1	<i>Requirements</i>	39
4.1.2	Architecture Modelling	40
4.1.3	Tools dan Teknologi	42
4.2	Perencanaan (<i>Planning</i>)	42
4.3	Implementasi (<i>Iteration to Release</i>)	45
4.3.1	<i>Analysis</i>	46
4.3.2	<i>Design</i>	47
4.3.3	<i>Coding</i>	48
4.3.4	<i>Whitebox Testing</i>	51
4.3.5	<i>Blackbox Testing</i>	54
4.4	<i>Productionizing Phase</i>	56
4.5	<i>Maintenance Phase</i>	58
4.6	<i>Death Phase</i>	58
	BAB V KESIMPULAN DAN SARAN	59
5.1	Kesimpulan	59

5.2 Saran	59
DAFTAR PUSTAKA	60
LAMPIRAN	64

DAFTAR TABEL

2.1	Perbandingan metodologi SDLC	12
2.2	Ranking Skor <i>Cyclomatic Complexity</i>	23
2.3	Fuel Consumption Rate Verification (PT Bisma Jaya, Fuel Consumption Rate Verification Document. Unpublished confidential document; 2021)	25
2.4	Contoh Laporan Penggunaan Bahan Bakar	26
2.5	Penelitian terdahulu mengenai <i>Internet of Things</i> (IoT)	27
3.1	Daftar <i>user story</i> rencana awal	35
4.1	Daftar <i>user story</i> Sistem Monitoring	39
4.2	Rencana Iterasi 1	42
4.3	Rencana Iterasi 2	43
4.4	Rencana Iterasi 3	43
4.5	Rencana Iterasi 4	44
4.6	Rencana Iterasi 5	45
4.7	Aturan Penomoran Kebutuhan Sistem	46
4.8	Pseudocode Halaman Engine Speed	49
4.9	Aturan Penomoran Whitebox Testing	51
4.10	<i>Whitebox Testing</i> API Engine Speed	53
4.11	Aturan Penomoran Blackbox Testing	54
4.12	<i>Blackbox Testing</i> Halaman Engine Speed	54
4.13	Umpam Balik Selama Pengembangan	56

DAFTAR GAMBAR

1.1	Kerangka Penelitian	6
2.1	Struktur Organisasi PT Bisma Jaya	8
2.2	Lapisan dan Komponen Arsitektur IoT (Sikder dkk. 2018)	9
2.3	Raspberry Pi dan 40 pin GPIO	11
2.4	Siklus Hidup Metode <i>Extreme Programming</i> (Anwer dkk. 2017)	16
2.5	Simbol pada ERD (Begum, 2015)	19
2.6	Contoh Notasi <i>Flowgraph</i> : (a) <i>if-else statement</i> (b) <i>while loop</i> (c) <i>loop with if statement break in the middle</i> (d) <i>loop with two entry points</i> (Sonarsource, 2020)	23
2.7	Contoh Test Case Blackbox Testing (Rahman 2018)	25
3.1	Diagram Alir Penelitian	33
3.2	Arsitektur Sistem Monitoring berbasis IoT	34
3.3	<i>Entity Relationship Diagram</i> (ERD) Sementara	36
3.4	<i>Wireframe</i> Halaman Engine Speed	37
4.1	Metafor Model Arsitektur Sistem Monitoring	41
4.2	Garis Besar Penggeraan Implementasi	46
4.3	Kebutuhan Fungsional Engine Speed	47
4.4	Frontend Halaman Engine Speed	48
4.5	Diagram Alir Halaman Engine Speed	50
4.6	Diagram Alir API Engine Speed	51
4.7	Notasi <i>Flowgraph Backend</i> Engine Speed	52
4.8	Frontend Halaman Overview	57
4.9	Frontend Halaman Home	57
4.10	Frontend Halaman OP41 Report	58
0.1	Validasi akurasi pembacaan alat oleh PT Bisma Jaya selaku mitra dan PT Pertamina Hulu Mahakam selaku pihak pengguna jasa	65

0.2	Pemasangan dan pengujian alat di lapangan pada salah satu armada yang beroperasi dibawah PT Bisma Jaya	65
0.3	Hasil Perhitungan Cyclomatic Complexity	66

DAFTAR LAMPIRAN

Dokumentasi	65
Hasil Implementasi	66

Halaman ini sengaja dikosongkan

BAB I

PENDAHULUAN

1.1 Latar Belakang

Indonesia merupakan negara maritim dengan luas laut dan perairan 62% (Wuryandani dan Meilani 2011). Potensi ini harus didukung berdasarkan prinsip *Blue Economy*. *Blue Economy* sendiri merupakan komponen penting dalam pengembangan keberlanjutan yang berfokus pada ekonomi maritim yang meliputi berbagai sektor seperti perikanan, akuakultur, dan transportasi maritim. Konsep *Blue Economy* berkaitan erat dengan *Sustainable Development Goal* ke 14 yang membahas mengenai pelestarian dan penggunaan lautan, laut, dan sumber daya laut secara berkelanjutan (LSE 2023). Menurut Perhubungan (2020), transportasi laut memegang peran strategis untuk mendorong pertumbuhan ekonomi di Indonesia. Salah satu bentuk untuk mendukung rencana jangka panjang ini adalah melalui upaya menuntaskan permasalahan fundamental dari industri tersebut. Pada industri maritim, salah satu biaya dengan rasio komposisi terbesar terletak pada biaya bahan bakar operasional. Tingginya biaya bahan bakar ini dapat memperlambat kemajuan industri maritim dikarenakan akan mengurangi pendapatan perusahaan, terlebih jika ternyata tingginya biaya bahan bakar ini disebabkan oleh hal lain diluar operasional. Sehingga, memastikan bahwa penggunaan bahan bakar lebih efisien dirasa perlu.

Efisiensi sendiri terbagi menjadi dua, yakni Efisiensi Teknologi dan Efisiensi Manajemen. Efisiensi Teknologi merujuk pada keterbaruan teknologi mesin yang mampu menghemat penggunaan bahan bakar dari waktu ke waktu. Sedangkan pada Efisiensi Manajemen memastikan bahwa bahan bakar sepenuhnya digunakan untuk mendukung operasi. Fokus pada penelitian ini adalah Efisiensi Manajemen. Untuk itu, diperlukan sebuah teknologi untuk melakukan validasi data penggunaan bahan bakar yang dilaporkan dengan nilai aktual yang dihabiskan.

Teknologi transformatif tersebut bernama IoT atau *Internet of Things* yang berpotensi merevolusi berbagai industri melalui kontrol dan pemantauan ekstensif secara jarak jauh (Hercog dkk. 2023). Kemampuan teknologi IoT dalam memberikan data secara jarak jauh membuka jalan bagi pelaku industri untuk merealisasi efisiensi

bahan bakar khususnya pada transportasi laut. Hal ini senada dengan penelitian yang dilakukan Suci dkk. (2023), yang menyatakan IoT memungkinkan integrasi mesin, sistem, dan proses untuk meningkatkan efisiensi operasional dan *predictive maintenance*.

Langkah untuk melakukan efisiensi dengan kontrol melalui teknologi IoT juga dinilai tepat mengingat minyak fosil akan habis di tahun 2070 (Review 2016) sehingga pelaku industri tidak hanya menghemat biaya operasional, tetapi secara tidak langsung juga menjaga lingkungan secara berkelanjutan. Dengan demikian, operasi akan dipastikan berjalan secara optimal dengan memanfaatkan bahan bakar secara maksimal. Sebaliknya, salah satu dampak yang ditimbulkan dari belum diterapkannya teknologi ini adalah kurangnya kontrol yang mengakibatkan celah pada pelanggaran hukum. Pada tahun 2020 terdapat kasus penggelapan bahan bakar yang mencapai 2.5 ton liter (Aditya 2022), sehingga menimbulkan kerugian negara mencapai 710 juta rupiah. Hal ini dapat diatasi menggunakan sistem monitoring berbasis IoT yang memungkinkan pemantauan secara jarak jauh. Sistem monitoring berbasis IoT yang dimaksud adalah suatu sistem yang menggunakan *Internet of Things* untuk memantau dan menyimpan data dari berbagai sensor. Dalam penelitian ini maka akan dikembangkan Sistem Monitoring berbasis IoT yang akan bekerja sama dengan salah satu perusahaan swasta yang bergerak di bidang transportasi laut sebagai mitra, yaitu PT Bisma Jaya.

PT Bisma Jaya merupakan perusahaan jasa maritim yang menyediakan berbagai jenis kapal untuk kebutuhan transportasi laut. Berdasarkan informasi yang diperoleh dari Direktur Operasional perusahaan, saat ini digunakan laporan harian dan bulanan sebagai acuan dalam mengestimasi jumlah bahan bakar yang diperlukan di bulan berikutnya. Permasalahan yang umum terjadi adalah waktu sampai yang lebih lama dari estimasi dan sulitnya mengontrol konsumsi bahan. Sistem Monitoring berbasis IoT dinilai cocok untuk menuntaskan permasalahan tersebut, dimana sistem memungkinkan pemantauan data kecepatan mesin dan bahan bakar secara jarak jauh agar sesuai dengan ketentuan yang berlaku, memastikan penggunaan bahan bakar termanfaatkan dengan maksimal.

Untuk merealisasi penelitian ini dibutuhkan lintas disiplin ilmu, yakni

elektronika dan komputer. Oleh karena itu, dibutuhkan suatu metode pengembangan sistem yang lebih menekankan kolaborasi serta komunikasi yang baik. Terdapat metode *Waterfall*, yang merupakan proses desain sekuensial yang digunakan dalam proyek pengembangan perangkat lunak. Ini mengikuti perkembangan linier melalui fase yang berbeda, termasuk pengumpulan dan analisis kebutuhan, desain, pengkodean, pengujian, dan pemeliharaan (Abbas 2016). Metode ini, mengasumsikan kebutuhan telah final di awal proyek, serta tiap fase harus diselesaikan terlebih dahulu sebelum lanjut ke fase berikutnya. Oleh karenanya, metode ini tidak cocok untuk diterapkan pada sistem yang memiliki kebutuhan yang dinamis. Sehingga, penggunaan metode *Waterfall* dirasa kurang cocok dalam pengembangan Sistem Monitoring berbasis IoT dikarenakan sulitnya untuk mengatur perubahan dan beradaptasi pada kebutuhan yang berkembang.

Agile merupakan pendekatan yang secara efektif dapat beradaptasi dengan kebutuhan yang berubah-ubah, yang mana ini sulit untuk diatur pada model *Waterfall*. Salah satu metode yang populer adalah Scrum. Scrum merupakan metodologi yang berfokus pada pengembangan berulang dan bertahap, fleksibilitas, dan perbaikan terus menerus. Metodologi ini sangat cocok untuk pengembangan proyek skala masif dengan personel pengembang yang banyak.

Lalu terdapat metode *Extreme Programming* (XP), sebuah metode *agile* yang menekankan pada kolaborasi, adaptasi, dan pengembangan iteratif (Matharu dkk. 2015). *Extreme Programming* metode yang ideal untuk digunakan tim skala kecil menengah dalam pengembangkan perangkat lunak dengan cepat serta fleksibel dalam menghadapi perubahan. Salah satu prinsip dari metode ini adalah keterlibatan pelanggan (Matharu dkk. 2015). Hal ini memastikan perangkat lunak memenuhi kebutuhan pelanggan dan mengurangi risiko pengembangan fitur yang tidak diperlukan.

Berdasarkan pertimbangan pilihan metode yang sudah dilakukan, metode yang paling cocok untuk diterapkan pada studi kasus penelitian ini adalah metode *Extreme Programming* (XP) karena dari perusahaan membutuhkan sistem yang dapat dengan cepat diimplementasikan tanpa harus melalui proses dokumentasi yang banyak. Metode ini cocok untuk pengembangan sistem dengan tim yang sedikit dan

dalam kurun waktu yang relatif singkat, serta bersifat fleksibel terhadap perubahan dikarenakan adanya kemungkinan perubahan kebutuhan terkait fitur-fitur yang ada pada sistem.

Diharapkan dengan adanya penelitian ini, Sistem Monitoring pada mesin diesel yang dikembangkan dapat membantu PT Bisma Jaya khususnya Direktur Operasional dalam melakukan pemantauan penggunaan bahan bakar serta menjaga efektivitas armada kapal yang sedang beroperasi yang mulanya melalui laporan harian/bulanan yang dibuat secara manual menjadi sistem yang dapat menyajikan data historis dan dapat diakses kapan saja.

1.2 Rumusan Masalah

Berdasarkan latar belakang dari penelitian, didapatkan rumusan masalah sebagai berikut.

1. Bagaimana sistem monitoring dirancang menggunakan metode *Extreme Programming*?
2. Bagaimana sistem monitoring dikembangkan menggunakan metode *Extreme Programming*?

1.3 Tujuan

Tujuan dari penelitian ini adalah sebagai berikut.

1. Untuk merancang sistem monitoring menggunakan metode *Extreme Programming*.
2. Untuk mengembangkan sistem monitoring menggunakan metode *Extreme Programming*.

1.4 Manfaat

Manfaat yang didapatkan pada penelitian ini adalah sebagai berikut.

1. Membantu perusahaan dalam melakukan pengawasan dan kontrol konsumsi bahan bakar armada kapal selama operasi.
2. Membantu perusahaan dalam memastikan efektivitas operasi

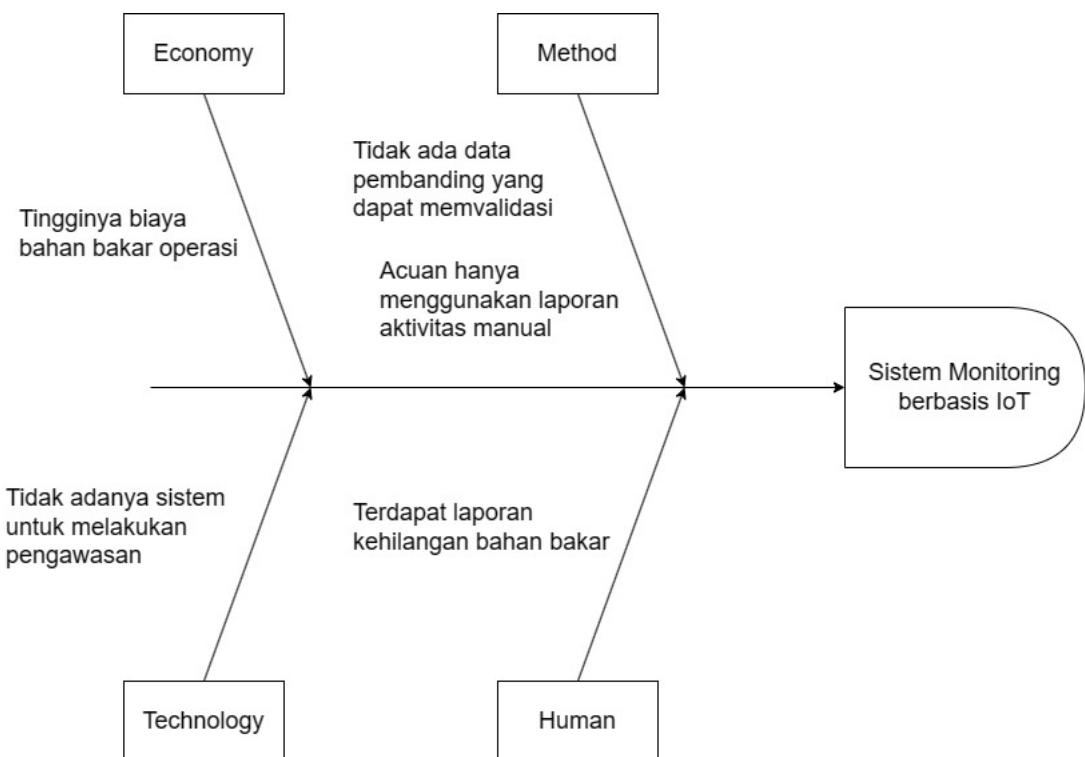
1.5 Batasan Penelitian

Batasan penelitian ini adalah sebagai berikut.

1. Fokus utama pada penelitian ini adalah pengembangan Sistem Monitoring berbasis web
2. Pada penelitian ini diterapkan 3 layer arsitektur IoT teratas: *App Layer*, *Data Processing Layer*, dan *Network Layer*
3. Sistem Monitoring berbasis IoT dikembangkan menggunakan framework NextJS, Django, dan MySQL sebagai *Database Management Systems (DBMS)*

1.6 Kerangka Pemikiran Penelitian

Berikut kerangka pikiran pada penelitian ini.



Gambar 1.1. Kerangka Penelitian

Gambar 1.1 merupakan kerangka pemikiran penelitian yang bertujuan untuk memberikan gambaran mengenai urgensi implementasi Sistem Monitoring berbasis *Internet of Things (IoT)* di PT Bisma Jaya. Perusahaan ini dihadapkan pada masalah utama berupa tingginya pengeluaran untuk bahan bakar selama operasionalnya, yang dipengaruhi oleh sejumlah permasalahan dalam aspek-aspek ekonomi, metode, teknologi, dan manusia.

Kategori ekonomi, laporan harian secara rutin dibuat setiap malamnya oleh tim operasional kapal. Selama operasi, mereka menjaga catatan aktivitas dalam sebuah jurnal yang mencatat waktu perjalanan dan berhenti kapal. Namun, terdapat kekurangan dalam pencatatan yaitu tidak adanya pencatatan jumlah jam operasi pada tiap kategori operasi tertentu. Akibatnya, ketika mereka membuat laporan, nilai *running hour* untuk tiap kategori operasi hanya dapat diestimasikan saja, yang bisa berpotensi mengakibatkan perhitungan konsumsi bahan bakar lebih tinggi dari seharusnya. Informasi lebih lanjut mengenai kategori operasi dapat dilihat pada Bab II bagian 2.14.

Kategori metode, perusahaan selama ini mengandalkan laporan bulanan yang

dibuat tim operasional kapal. Hanya saja, tidak ada data faktual yang dapat dijadikan pembanding terhadap laporan yang dibuat. Selain itu, laporan tersebut baru diterima setiap akhir bulan, sehingga perusahaan tidak memiliki data apapun hingga mendapat temuan dari pihak pengguna.

Kategori teknologi, tidak adanya suatu sistem monitoring yang dipasang pada armada juga menjadi salah satu masalah utama perusahaan. Selama ini, perusahaan hanya mengandalkan data dari AVTS untuk mengetahui data *running hour*, kecepatan (knot), dan posisi kapal. Tetapi, alat ini tidak dapat memberikan informasi detail terkait kecepatan mesin dan konsumsi bahan bakar.

Kategori manusia, pengguna jasa perusahaan telah mengalami situasi di mana mereka melaporkan adanya kejadian kehilangan bahan bakar. Sebelum armada kapal mengisi ulang bahan bakarnya, operator *fuel management* pengguna jasa akan melakukan pengukuran yang dikenal dengan istilah "sounding." Hasil dari pengukuran ini akan dibandingkan dengan laporan harian yang disusun oleh tim operasional kapal. Apabila ditemukan selisih sebesar lebih dari 100 liter, operator *fuel management* akan melaporkan indikasi kehilangan yang dapat mengakibatkan dikenakan denda.

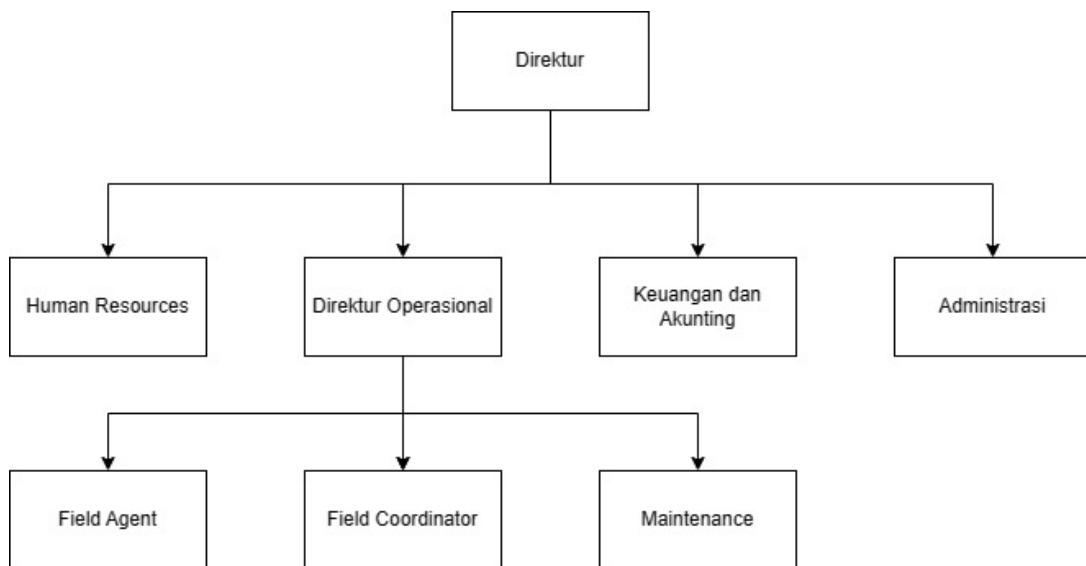
Secara garis besar, didapatkan inti permasalahan yang terjadi di PT Bisma Jaya adalah belum adanya langkah kontrol pada bahan bakar pada kapal yang dimiliki, sehingga pada penelitian ini akan dikembangkan Sistem Monitoring mesin diesel yang akan membantu perusahaan dalam melakukan pemantauan jumlah bahan bakar selama operasi.

BAB II

TINJAUAN PUSTAKA

2.1 PT Bisma Jaya

PT Bisma Jaya merupakan perusahaan yang bergerak di industri transportasi angkutan laut yang berbasis di Balikpapan, Kalimantan Timur. Sejak tahun 2011, perusahaan telah menyediakan berbagai jenis kapal untuk kebutuhan transportasi industri. Dalam menjalankan tugasnya PT Bisma Jaya memiliki struktur organisasi seperti pada Gambar 2.1.



Gambar 2.1. Struktur Organisasi PT Bisma Jaya

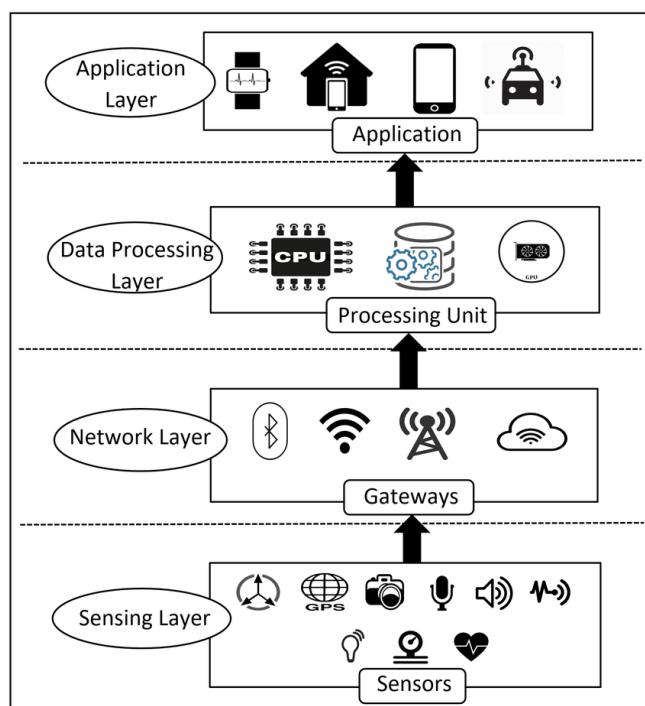
Gambar 2.1 memberikan gambaran struktur organisasi dari PT Bisma Jaya yang dipimpin oleh Direktur yang membawahi *Human Resource*, Direktur Operasional, Keuangan dan Akunting, dan Administrasi. Direktur Operasional membawahi beberapa bagian seperti *Field Agent*, *Field Coordinator*, dan *Maintenance*. Seluruh kegiatan operasional kapal berada dalam tanggung jawab Direktur Operasional yang memastikan seluruh operasi berjalan dengan lancar serta membuat keputusan strategis dalam mengelola biaya operasional.

Hasil luaran pada penelitian ini akan secara spesifik akan mendukung Direktur Operasional dalam membuat keputusan. Diharapkan Sistem Monitoring

yang dikembangkan dapat memberikan wawasan mengenai performa armada kapal sekaligus menjadi acuan dalam melakukan kontrol bahan bakar.

2.2 Internet of Things

Internet of Things (IoT) merujuk pada keterhubungan antara obyek, perangkat, mesin satu dengan lainnya dan internet mengizinkan mereka untuk mengumpulkan dan menukar data (Gazis 2021). Secara arsitektur IoT dapat dibagi menjadi 4 lapisan utama: *sensing layer*, *network layer*, *data processing layer*, dan *application layer* (Sikder dkk. 2018). Detailnya dapat dilihat pada Gambar 2.2



Gambar 2.2. Lapisan dan Komponen Arsitektur IoT (Sikder dkk. 2018)

Berikut gambaran umum pada setiap lapisan:

1. *Sensing Layer*

Lapisan ini bertanggung jawab untuk memanfaatkan berbagai sensor dan perangkat untuk mengumpulkan data. Sensor seringkali memberikan data berupa angka mentah seperti tegangan atau sinyal. Oleh karena itu, perangkat IoT dapat memprosesnya terlebih dahulu sebelum dikirim ke server - disebut juga

dengan *edge-computing* - atau langsung meneruskan data tersebut ke lapisan jaringan untuk diproses di server.

2. *Network Layer*

Lapisan ini bertugas mengirimkan data yang diperoleh sensor ke lapisan pemrosesan data untuk diolah. Lapisan ini juga bertugas mengawasi bagaimana perangkat jaringan IoT berkomunikasi satu sama lain melalui protokol komunikasi yang ditetapkan. Untuk menjaga keamanan komunikasi, digunakan token autentikasi setiap adanya pengiriman data ke *server*.

3. *Data Processing Layer*

Pemrosesan dan analisis data sensor berada di bawah lingkup lapisan ini. Selain itu, lapisan ini juga bertugas dalam mengelola dan menyimpan data. Lapisan pemrosesan data sangat penting untuk menghasilkan wawasan berharga dan mengambil tindakan yang sesuai berdasarkan data yang dikumpulkan.

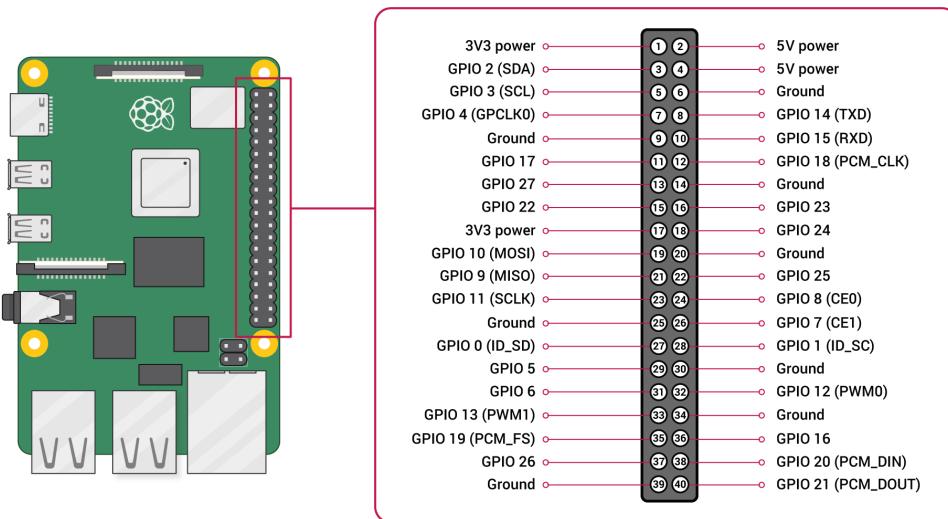
4. *Application Layer*

Dengan menggunakan data yang dikumpulkan dan diproses, lapisan ini bertanggung jawab untuk memberikan *actionable insight* kepada pengguna akhir melalui sistem atau aplikasi. Sistem menampilkan data dari server dalam bentuk grafik maupun tabel dan dapat diakses dari berbagai perangkat.

2.3 Raspberry Pi

Raspberry Pi merupakan *single-board computer* (SBC) yang telah mendapatkan perhatian dan popularitas yang signifikan dalam beberapa tahun terakhir. Teknologi inovatif ini memungkinkan berbagai penerapan dan sekarang penting dalam bidang ilmu dan teknik komputer (Johnston dan Cox 2017). Raspberry Pi adalah pilihan yang bagus untuk aplikasi *Internet of Things* (IoT) karena portabilitasnya, paralelismenya, keterjangkauannya, dan konsumsi dayanya yang rendah (Hosny dkk. 2023). Hal yang membuat Raspberry Pi dapat diandalkan sebagai perangkat IoT adalah adanya 40 pin GPIO yang memungkinkan ia dihubungkan ke beragam sensor

dengan berbagai *interface*. Raspberry Pi serta informasi GPIO dapat dilihat pada Gambar 2.3.



Gambar 2.3. Raspberry Pi dan 40 pin GPIO

Setiap pin GPIO dapat digunakan sebagai pin input maupun output, dan dapat digunakan untuk berbagai kebutuhan. Terdapat pin 5v dan 3.3v yang berjumlah masing-masing 2, juga beberapa pin *ground* yang tidak dapat dikonfigurasi. Sisanya merupakan pin *general purpose* 3.3v, yang berarti output diatur ke 3.3v dan input toleran dengan nilai 3.3v. Pin output dapat diatur ke *high* (3.3v) dan *low* (0v). Begitu juga dengan pin input, dapat membaca *high* (3.3v) dan *low* (0v). Selain itu, pin GPIO juga dapat digunakan untuk kebutuhan yang memerlukan jenis pin yang spesifik seperti *PWM* (*pulse-width modulation*) untuk membuat sinyal analog; *SPI* (*serial peripheral interface*) untuk transfer data antar Raspberry Pi dengan perangkat periferal; *I2C* (*inter-integrated circuit*) untuk komunikasi dengan berbagai jenis sensor; dan Serial untuk pembacaan data serial.

2.4 Perbandingan SDLC

Berikut adalah perbandingan dari metodologi *Extreme Programming* dengan salah satu metode *sequential*, yaitu *Waterfall* dan metode *agile* lainnya, yaitu *Scrum* menurut Fahrurrozi dan Azhari (2013) dan Suryantara dan Andry (2018).

Tabel 2.1. Perbandingan metodologi SDLC

Tahapan dalam pengembangan	<i>Extreme Programming</i>	<i>Waterfall</i>	<i>Scrum</i>
<i>Planning</i>	Pada Tahap ini Tahap ini merupakan dibagi menjadi 2 langkah awal dimana bagian: <i>Release Planning</i> kebutuhan proyek dikumpulkan dan <i>Iteration Planning</i> . <i>Release Planning</i> dianalisis. <i>Planning</i> dilakukan untuk mengumpulkan kebutuhan sistem dalam bentuk <i>user story</i> . Lalu, pada <i>Iteration Planning</i> <i>user story</i> tersebut dikonversi menjadi <i>task</i> yang dapat dikerjakan dan diurutkan berdasarkan prioritas dan tingkat kesulitannya.	Tahap ini merupakan Tahap ini menjadi 2 bagian: <i>Sprint Planning</i> dan <i>Release Planning</i> . <i>Sprint Planning</i> dilakukan setiap awal sprint dan melibatkan tim untuk memilih pekerjaan yang akan mereka selesaikan di sprint tersebut. <i>Release Planning</i> dilakukan setiap awal rilis dan melibatkan tim merencanakan fitur yang akan dimasukkan ke rilis tersebut.	Tahap ini dibagi menjadi 2 bagian: <i>Sprint Planning</i> dan <i>Release Planning</i> .
<i>Analysis</i>	Pada tahap ini dibuat kebutuhan fungsional yang mengacu pada <i>user story</i> yang telah dibuat.	Pada Tahap ini kebutuhan yang dikumpulkan akan dipecahkan menjadi potongan yang dapat dikelola.	Analisis terjadi saat Sprint Planning, dimana tim akan memilih perkerjaan yang akan mereka selesaikan di sprint tersebut

Tahapan dalam pengembangan	<i>Extreme Programming</i>	<i>Waterfall</i>	<i>Scrum</i>
<i>Design</i>	Tahap ini merupakan dimana desain sederhana dibuat dan akan disempurnakan seiring berjalannya iterasi. Desain setidaknya dibuat dalam bentuk <i>wireframe</i> sebagai acuan ketika implementasi.	Tahap desain melibatkan pembuatan rencana rinci untuk perangkat lunak berdasarkan persyaratan yang dibuat dikumpulkan dalam fase perencanaan dan analisis.	Perancangan selama <i>Sprint Planning</i> , dimana tim memilih pekerjaan yang akan mereka selesaikan selama sprint.
<i>Implementation</i>	Pengembang bekerja dalam waktu singkat untuk menghasilkan perangkat lunak yang berfungsi, dan berkembang seiring berjalannya iterasi.	Tahap implementasi melibatkan koding perangkat lunak berdasarkan rencana rinci yang dibuat pada fase desain.	Implementasi selama sprint, dimana tim menyelesaikan pekerjaan yang mereka pilih selama <i>Sprint Planning</i> .

Tahapan dalam <i>Extreme Programming</i>	<i>Waterfall</i>	<i>Scrum</i>
<i>Support & Security</i>	adalah proses yang berlangsung sepanjang proyek. Tahap ini akan berakhir setelah memasuki <i>Death Phase</i>	Tahap <i>support & security</i> terjadi setelah software dikirimkan. Tahap ini melibatkan pemeliharaan <i>Death</i> dan <i>pembaruan perangkat lunak</i> untuk memastikannya terus memenuhi kebutuhan pengguna.

Secara garis besar, *Software Development Life Cycle* memiliki lima tahapan, yaitu *Systems Planning*, *Systems Analysis*, *Systems Design*, *Systems Implementation*, dan *System Support and Security*. Metode *Sequential* dan *Agile* memiliki cara implementasi yang cukup berbeda. Pada metode sekuensial seperti *Waterfall*, perangkat lunak dikembangkan secara linear. Artinya, sebelum tahap selanjutnya, tahap sebelumnya sudah harus diselesaikan. Adanya perbedaan atau perubahan mengharuskan pengembang untuk kembali ke tahap awal. Sedangkan, pada metode *Agile* seperti *Extreme Programming (XP)* atau *Scrum*, aktivitas di tahapan-tahapan tersebut dapat secara dinamis berubah dan akan menyesuaikan kembali di tahapan sebelumnya. Oleh karenanya, metode sekuensial tidak akan dipilih pada penelitian ini.

Lebih lanjut, komparasi antara metode *Agile* yaitu *Extreme Programming (XP)* dengan *Scrum* pada fase perencanaan. Pada XP, perencanaan dilakukan menjadi dua bagian, yakni *Release Planning* dan *Iteration Planning*. Tujuan *Release Planning* adalah untuk mengetahui fitur apa saja yang diperlukan sistem dan kapan fitur tersebut dikerjakan. Lalu, terdapat *Iteration Planning* yang dilakukan setiap awal

iterasi. Pada fase ini, pengembang menyiapkan rencana untuk mengimplementasi fitur yang pada rilis saat itu. Kemudian kebutuhan sistem akan dibuat menjadi *task* berdasarkan *user story* yang dibuat. Urutan pengerjaan akan dikelompokkan menjadi Iterasi yang berisi kumpulan *user story* yang telah diprioritaskan. Pengerjaan teknis yang dilakukan selama iterasi meliputi analisis, desain sederhana, pengkodean, dan testing. Sedangkan untuk *Scrum*, perencanaan dibagi menjadi dua bagian yakni *Sprint Planning* dan *Release Planning*. Singkatnya, *Release Planning* berisi seluruh fitur yang akan dikembangkan dan pengerjaannya dibuat menjadi *backlog* yang kemudian akan dibagi di *Sprint Planning* di setiap awal *sprint*.

Pada tahap implementasi, XP dan *Scrum* sebenarnya tidak begitu jauh berbeda karena mengadopsi pendekatan yang sama. Namun, terdapat beberapa perbedaan utama diantara keduanya. Pertama, satu iterasi (yang disebut dengan *sprint*) pada *Scrum* dikerjakan selama 2 pekan hingga 1 bulan. XP mengerjakan satu iterasi lebih singkat yakni hanya 1 hingga 2 pekan saja; Kedua, *sprint* tidak boleh diubah selesai ditetapkan ketika *sprint planning*. Komitmen harus dipegang untuk menyelesaikan item backlog. XP lebih fleksibel terhadap perubahan dalam iterasi selama fitur tersebut belum dikerjakan. *Task* dengan ukuran sebanding dapat ditukar ke iterasi sebagai ganti dari fitur yang belum dimulai; Ketiga, pengerjaan fitur pada XP ditentukan oleh urutan prioritas. Sedangkan pada Scrum, prioritas *backlog* tidak menjadi acuan dalam penentuan item yang akan dikerjakan di suatu *Sprint*, karena bisa jadi pengembang yang akan mengerjakan fitur tersebut harus fokus dengan item yang memiliki prioritas yang tinggi pada *sprint* yang sama.

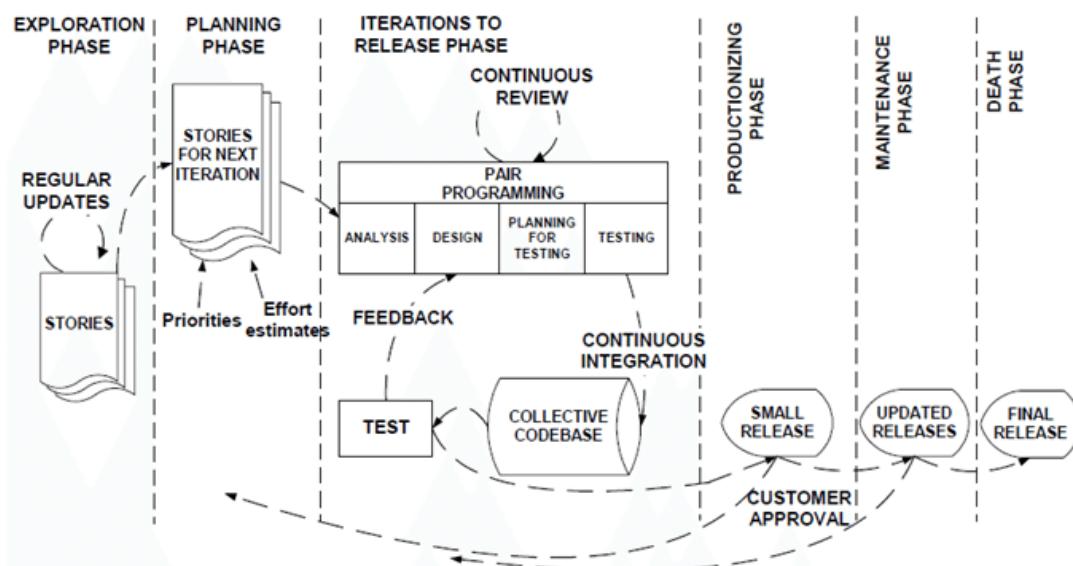
Pada akhirnya, dalam penelitian ini dipilihlah metode *Extreme Programming* (XP) dengan beberapa pertimbangan seperti perbaikan atau penambahan fitur baru ditengah fase *iteration to release*, juga dari hasil jurnal dengan judul serupa. Riset yang dilakukan Rumandan (2023), berfokus pada pengembangan sistem informasi *customer service* menggunakan metode *Extreme Programming (XP)* dimana ia menggaris bawahi kemampuan XP dalam memungkinkan interaksi pengguna secara langsung sehingga dapat segera menangani informasi dan resolusi komplain. Matharu dkk. (2015) mendeskripsikan XP sebagai metodologi yang mendukung pengembangan perangkat lunak secara iteratif oleh tim skala kecil, sehingga meningkatkan

produktivitas dan kualitas perangkat lunak untuk memenuhi kebutuhan yang terus berkembang.

2.5 *Extreme Programming (XP)*

Extreme Programming (XP) adalah pendekatan *agile software development* yang memberikan penekanan pada kerja sama, pengembangan iteratif dan berulang, serta kemampuan beradaptasi terhadap perubahan kebutuhan. XP merupakan metodologi sederhana yang dibuat untuk tim pengembang kecil yang bertujuan untuk meningkatkan kualitas dan produktivitas perangkat lunak (Matharu dkk. 2015). Kesulitan yang ditimbulkan oleh siklus pengembangan yang panjang dalam praktik pengembangan perangkat lunak konvensional menyebabkan terciptanya XP (G. Rao, Krishna, dan K. Rao 2013).

Ada berbagai prinsip dasar yang mendefinisikan XP. *Continuous planning*, yang memerlukan komunikasi dan kolaborasi rutin antara pengembang dan pemangku kepentingan untuk memastikan bahwa tujuan dan persyaratan proyek dipahami dan dipenuhi (Matharu dkk. 2015). Siklus hidup pada metode *Extreme Programming* meliputi *Exploration Phase*, *Planning Phase*, *Iteration to Release Phase*, *Productionizing Phase*, *Maintenance Phase*, dan *Death Phase*. Lebih lengkapnya dapat dilihat pada gambar berikut.



Gambar 2.4. Siklus Hidup Metode *Extreme Programming* (Anwer dkk. 2017)

Berikut penjelasan detail untuk setiap fase:

1. ***Exploration Phase***: Pada fase ini, dihasilkan *user story* yang dibuat berdasarkan hasil pengambilan data baik dari observasi, interview, dan dialog dengan mitra. *User story* ini dapat bertambah seiring waktu mengikuti kebutuhan mitra. Disamping itu, *wireframe* dan *Entity Relationship Diagram* juga dibuat berdasarkan data yang diperoleh dari *user story*.
2. ***Planning Phase***: Selanjutnya, *user story* yang sebelumnya dibuat akan dikumpulkan dan diprioritaskan berdasarkan perhitungan poin story. Ini akan membantu kita dalam menentukan *user story* mana yang akan dikerjakan pada iterasi berikutnya.
3. ***Iteration to Release Phase***: Tahap iterasi merupakan tahap dimana pengembang akan mengimplementasi sistem berdasarkan *user story* yang ditentukan. Pertama, dilakukan tahap analisis untuk mengonversi kebutuhan mitra menjadi kebutuhan fungsional sistem. Lalu, dilakukan desain tampilan sederhana berdasarkan *wireframe* yang telah dibuat. Terakhir, dilakukan pengujian oleh pengembang sebelum kode diunggah ke repositori. Proses *programming* dilakukan secara parallel dari tahap analisis hingga pengujian. Setelah sistem berhasil melewati *unit test*, sistem akan diuji oleh mitra dan hanya dapat lanjut ke tahap berikutnya setelah mendapatkan persetujuan.
4. ***Productionizing Phase***: Sistem yang telah diunggah di repositori akan diluncurkan di server dengan mode pengembang. Ini memungkinkan mitra untuk melakukan pengujian fitur yang masih dalam proses persetujuan serta memberikan umpan balik secara berkala.
5. ***Maintenance Phase***: Iterasi yang mendapatkan persetujuan selanjutnya akan diluncurkan pada tahap ini. Dapat dikatakan sistem yang terdapat pada tahap ini merupakan gambaran terakhir dari sistem secara keseluruhan.
6. ***Death Phase***: Ini merupakan tahap terakhir dimana sistem akan diluncurkan secara penuh di server dengan mode produksi.

Melihat dari tantangan industri mitra yang dinamis serta perlunya kolaborasi yang kuat dari berbagai lintas disiplin ilmu untuk mewujudkan Sistem Monitoring berbasis IoT ini, diputuskanlah metode *Extreme Programming* sebagai metodologi pengembangan perangkat lunak yang menekankan pada komunikasi dan kolaborasi serta sifatnya yang *agile* memungkinkan pengembang untuk menjawab berbagai tantangan industri tanpa interupsi selama proses pengembangan sistem.

2.6 *User Story*

User story merupakan deskripsi singkat sebuah fitur dari perspektif pengguna akhir, biasanya ditulis dengan format seperti berikut 'As WHO, I want WHAT, so that WHY' (Dwitama dan Rusli 2020). Menurut Raharjana, Siahaan, dan Faticah (2019) *User story* merupakan bagian fundamental dari pengembangan *agile* dengan membantu pengembang dalam memahami kebutuhan pengguna dengan efektif. *User story* sendiri tidak hanya digunakan untuk melakukan pengumpulan kebutuhan tetapi juga dapat dikonversi menjadi *automated test case* sehingga mempercepat proses pengujian.

Pada metode *Extreme Programming*, *user story* memainkan peran penting pada *Exploration Phase* dimana hasil dari *user story* sendiri akan menjadi task yang dapat dikerjakan oleh pengembang. Selain berisi deskripsi singkat, pada fase selanjutnya *user story* nantinya akan memiliki tingkat prioritas serta estimasi waktu untuk menentukan iterasi *task*.

2.7 *Entity Relationship Diagram*

Entity relationship diagram basis data direpresentasikan secara visual dalam diagram hubungan entitas (ERD). Dengan menggunakan metode *top-down*, ini mewakili hubungan antara entitas dan atributnya dan mengatur data berdasarkan informasi semantik (Chen 1976). Untuk memberikan representasi yang jelas dan ringkas tentang struktur dan hubungan dalam basis data, ERD sering digunakan dalam desain dan pemodelan basis data (Supriyadi, Andryana, dan Gunaryati 2022). Simbol atau notasi ERD dapat dilihat pada Gambar dibawah.

ERD memiliki tiga komponen utama sebagai notasi. Pertama terdapat entitas,

Name	Symbol	Meaning
Entity		Object with the properties and the functions
Attribute		Behavioral values of entities
Relationship		Relationships among the database components
Links		Connectivity between the entities, relationships and the attributes

Gambar 2.5. Simbol pada ERD (Begum, 2015)

yang merupakan objek yang bersifat konkret maupun abstrak. Pada penerapan di MySQL, objek dapat dituangkan menjadi tabel; Kedua, atribut/*field* untuk mendeskripsikan karakteristik suatu entitas. Ini akan diimplementasi menjadi kolom dari masing-masing tabel; Terakhir, relasi yang menggambarkan hubungan antar entitas. Misal, Perusahaan memiliki Kapal atau Pengguna merupakan bagian dari Perusahaan. Relasi ini kemudian dipetakan bagaimana data berhubungan satu sama lainnya yang terbagi menjadi empat, yaitu:

1. *One to One* (1:1): Setiap Entitas A dapat berhubungan paling banyak dengan satu pada himpunan Entitas B. Misal, satu Kapal hanya dapat memiliki satu Konfigurasi.
2. *One to Many* (1:M): Setiap Entitas A dapat berhubungan lebih dari satu anggota Entitas B. Misal, satu Perusahaan dapat memiliki banyak Kapal.
3. *Many to One* (M:1): Ini merupakan kebalikan dari relasi One to Many. Misal,

banyak Pengguna dapat mengakses satu Kapal.

4. *Many to Many* (M:N): Setiap entitas pada kumpulan Entitas A dapat berhubungan dengan banyak entitas pada kumpulan Entitas B. Misal, banyak User dapat mengakses banyak Kapal.

2.8 MySQL

MySQL adalah sistem manajemen *database open-source* yang umum digunakan sebagai penghubung perangkat lunak dengan *database* server. MySQL dapat secara efektif mengelola banyak pengguna secara bersamaan dan data dalam jumlah masif (Gómez-Hernández dkk. 2023). Kemampuan kueri yang cukup kuat dan banyaknya dokumentasi menjadi pertimbangan pemilihan basis data pada sistem yang akan dibangun.

Dalam penelitian ini, MySQL akan digunakan sebagai tempat menyimpan metadata sekaligus data yang dikumpulkan dari sensor. Sistem Monitoring dapat berinteraksi dengan *database* yang dimungkinkan oleh API atau *Application Programming Interface*.

2.9 Application Programming Interface

Application Programming Interface (API) merupakan sekumpulan aturan dan protocol yang memungkinkan perangkat lunak berbeda untuk berkomunikasi satu dengan lainnya (Raatikainen et al, 2021). Pada pengembangan sistem berbasis web, API dilayani (*serve*) dalam bentuk tautan atau disebut juga dengan *endpoint* yang memiliki jenis *request* seperti POST, GET, UPDATE, PUT, dan DELETE sehingga klien dapat menyisipkan data atau *payload* ketika melakukan permintaan. Data ini kemudian diolah di server dan diteruskan ke basis data untuk disimpan.

Dalam penelitian ini, tiap *endpoint* akan selalu dilengkapi dengan token autentikasi agar hanya klien yang memiliki otorisasi yang dapat mengirim permintaan ke server. Tiap perangkat IoT yang dipasang akan dilengkapi dengan *authentication key* masing-masing untuk mengidentifikasi sumber perangkat. API digunakan agar memungkinkan perangkat IoT yang dipasang di kapal dapat mengirim data yang

diperoleh ke server melalui *endpoint* yang sudah didefinisikan.

2.10 Django

Django merupakan *framework* Python dalam pengembangan web. Python adalah bahasa pemrograman populer yang digunakan secara luas di berbagai bidang. Ia terkenal dengan syntaxnya yang singkat dan sederhana, sehingga cocok untuk otomatisasi proses dan pengintegrasian aplikasi (Bühler dkk. 2022). Popularitas Python dapat dikaitkan dengan kemampuan beradaptasi dan ketersediaan berbagai *library* dan *framework* yang mempercepat dan menyederhanakan pengembangan (Malloy dkk. 2017).

Framework ini akan digunakan untuk menyediakan *Application Programming Interface* (API) agar perangkat IoT dapat mengirim data ke server. Salah satu alasan kuat digunakannya Django adalah menjaga konsistensi tipe data yang dikirim oleh perangkat IoT. Selain itu, adanya *library* Python seperti Numpy memungkinkan kita untuk mengelola data berukuran masif dengan waktu relatif lebih cepat sehingga ketika pemrosesan data tidak memakan waktu lama.

2.11 NextJS

NextJS merupakan *framework* Javascript yang menjadi standar dalam pengembangan web modern berbasis JavaScript. JavaScript sendiri merupakan bahasa pemrograman yang sering digunakan khususnya pada pemrograman web. Menurut Tomasdottir, Aniche, dan Deursen (2020). Umumnya web dengan *framework* JavaScript dimuat disisi klien atau browser pengguna, disebut juga dengan *Client Side Rendering* (CSR). Namun, NextJS memungkinkan web untuk dimuat di server terlebih dahulu, disebut juga dengan *Server Side Rendering* (SSR) sehingga web dapat lebih cepat untuk dimuat.

Dalam penelitian ini, NextJS digunakan dalam pengembangan *frontend* atau tampilan dari sistem. Hal ini digunakan untuk meningkatkan pengalaman pengguna dengan kombinasi fitur CSR dan SSR yang dimiliki NextJS untuk mengelola data serta tampilan dari sistem.

2.12 Whitebox Testing

Whitebox testing, dikenal juga sebagai *structural testing* merupakan sebuah metode untuk memeriksa struktur internal dari perangkat lunak menjadi desain *test case* berdasarkan struktur kontrol desain (Nidhra dan Dondeti 2012). *Whitebox testing* merupakan pengujian yang dilakukan untuk menguji perangkat lunak dengan cara menganalisis struktur internal dan kode perangkat lunak. Pengujian ini berfokus pada aliran/*flow* input dan output dari perangkat lunak. Berikut teknik-teknik struktural pada *Whitebox testing*:

1. Control flow/Coverage testing

- *Statement coverage*
- *Branch coverage*
- *Decision/condition coverage*
- *Function coverage*

2. Basis path testing

- *Flow graph notation*
- *Cyclomatic complexity*
- *Deriving test case*
- *Graph matrices*

3. Loop testing

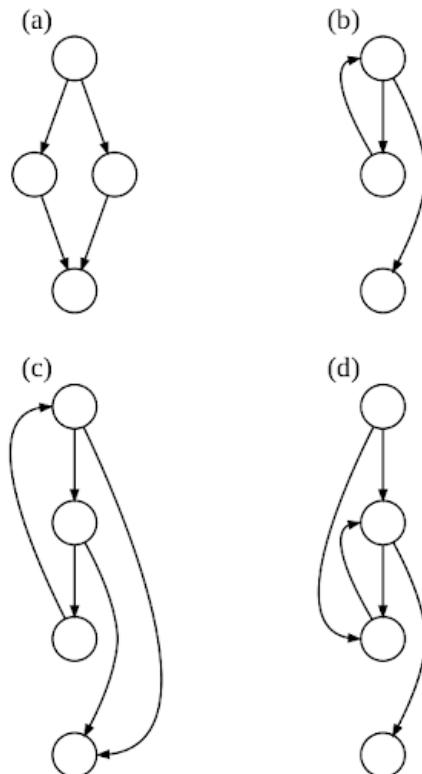
- *Simple loop*
- *Nested loop*
- *Concatenated loop*
- *Unstructured loop*

4. Data flow testing

Tabel 2.2. Ranking Skor *Cyclomatic Complexity*

Skor	Ranking dan Keterangan
1-5	A (<i>low risk - simple block</i>)
6-10	B (<i>low risk - well structured and stable block</i>)
11-20	C (<i>moderate risk - slightly complex block</i>)
21-30	D (<i>more than moderate risk - more complex block</i>)
31-40	E (<i>high risk - complex block, alarming</i>)
41+	F (<i>very high risk - error-prone, unstable block</i>)

Pada penelitian ini, digunakan salah satu strategi pengujian yakni *Basis Path Testing* oleh Watson, Wallace, dan McCabe (1996) menggunakan perhitungan *Cyclometric Complexity (CYC)*. Perhitungan ini dilakukan untuk mengukur tingkat kompleksitas kode serta memberikan informasi jumlah *test case* yang perlu dibuat agar seluruh blok dapat dilewati ketika melewati fase pengujian. Perhitungan CYC diawali dengan melakukan observasi pada *flowgraph*—sebuah representasi grafis dari setiap jalur yang akan dilewati program ketika dieksekusi.



Gambar 2.6. Contoh Notasi *Flowgraph*: (a) *if-else statement* (b) *while loop* (c) *loop with if statement break in the middle* (d) *loop with two entry points* (Sonarsource, 2020)

Hasil perhitungan didapatkan menggunakan rumus yang dapat dilihat pada Notasi 2.1. Melihat dari ilustrasi *flowgraph* pada Gambar 2.6, jumlah tanda panah direpresentasikan sebagai E (edges) dan jumlah lingkaran sebagai N (nodes). Terakhir, didapatkan tingkat kompleksitas kode berdasarkan rentang skor pada Tabel 2.2.

$$V(G) = E - N + 2 \quad (2.1)$$

Pengujian *Whitebox* pada akhirnya digunakan guna memastikan bahwa output dari data yang telah diproses telah sesuai dengan yang diharapkan. Contoh, ketika melakukan konversi log data menjadi nilai *running hour* sistem harus memastikan penjumlahan data tersebut akan menghasilkan output jam dan menit. Oleh karenanya, pengujian ini sangat penting guna menghasilkan data yang dapat diandalkan ketika diproses di '*Data Processing Layer*' sebelum diantar ke '*App Layer*'.

2.13 *Blackbox Testing*

Blackbox testing, juga dikenal sebagai *functional testing* merupakan sebuah metode pengujian untuk memverifikasi bahwa sistem telah bekerja dengan semestinya berdasarkan kebutuhan yang telah ditetukan (Noerlina, Chandra, dan Mursitama 2020). Pada metodologi XP, tim pengembang dapat memastikan perangkat lunak telah memenuhi ekspektasi pengguna dan fungsi secara benar tanpa perlu mengetahui bagaimana sistem internal bekerja yang dimana ini penting untuk memvalidasi behavior perangkat lunak berdasarkan input dan output tanpa menyelidiki kode internal. Contoh *test case* pada *Blackbox testing* dapat dilihat pada Gambar 2.7.

Dalam penelitian ini, *blackbox testing* akan dilakukan setelah kode diunggah ke *collective codebase* secara bertahap sebagai proses persetujuan mitra. Umpaman balik dari mitra akan menentukan apakah task akan dikerjakan sesuai dengan iterasi yang direncanakan atau di iterasi selanjutnya.

Test Code	Test Case	Test Steps	Expected Result	Actual Result	Pass/Fail
Test 1	Check Facebook login functionality with logged in Facebook user	1. Go to registration page 2. Click on “Login with Facebook” button	The window is redirected to Facebook, then to homepage with showing the Facebook timeline name on the top right of the bar. The name and details also saved in database.	As expected	Pass

Gambar 2.7. Contoh Test Case Blackbox Testing (Rahman 2018)

2.14 Metode Perhitungan Bahan Bakar

Dokumen Fuel Consumption Rate Verification (FCRV) digunakan sebagai salah satu acuan/bentuk kontrol bahan bakar kapal antara mitra dengan pengguna jasa dan hanya digunakan selama operasi di wilayah pengguna jasa. Dokumen ini berisi informasi kategori operasi berdasarkan rentang kecepatan mesin tertentu. Ini akan digunakan awak kapal ketika hendak melaporkan jumlah konsumsi bahan bakar berdasarkan jumlah running hour pada rentang angka kecepatan yang telah ditentukan. Berikut contoh isi dari Dokumen FCRV.

Tabel 2.3. Fuel Consumption Rate Verification (PT Bisma Jaya, Fuel Consumption Rate Verification Document. Unpublished confidential document; 2021)

Operation Category	Max Fuel Used (L)	RPM	Average Speed (knot)
Full Speed	28	1100	5
Economical Speed	18	900-1000	4
Slow Speed/Maneuver	11	700-800	3
Idle Speed	6	600	0
Standby (M/E Off)	0	0	0

Pada tabel diatas, terdapat 4 kategori operasi yakni *Full Speed*, *Economical Speed*, *Slow Speed/Maneuver*, dan *Idle Speed*. Full Speed adalah kondisi kecepatan mesin tertinggi yang hanya digunakan di laut lepas, nilai maksimum konsumsi bahan

bakar (FCR) dalam 1 jam mencapai 28L. Lalu, terdapat *Economical Speed*. Ini merupakan kategori kecepatan tertinggi kedua dan yang paling sering digunakan ketika menyusuri sungai. Kategori ini memiliki rentang RPM 900-1000 dengan nilai FCR 18L dalam 1 jam. Selanjutnya terdapat *Slow Speed*, dimana kecepatan ini digunakan untuk mengatur posisi kapal di pelabuhan. Kategori ini memiliki rentang RPM 700-800 dengan nilai FCR 11L dalam 1 jam. Terakhir, *Idle Speed* dimana kapal dalam kondisi tidak bergerak namun mesin menyala. Rentang RPM pada kategori ini adalah 700 kebawah dan hanya memakan bahan bakar 6L dalam 1 jam.

Pada praktiknya, awak kapal hanya mengisi nilai *running hour* dari masing-masing kategori untuk mendapatkan nilai konsumsi bahan bakar. Nilai *running hour* ini sebelumnya didapatkan berdasarkan estimasi mengikuti jurnal aktivitas/pergerakan kapal. Setelah dipasang perangkat IoT di kapal, awak kapal dapat langsung mengikuti nilai *running hour* berdasarkan tiap kategori yang ditampilkan di sistem. Contoh pengisian tabel pada laporan adalah sebagai berikut.

Tabel 2.4. Contoh Laporan Penggunaan Bahan Bakar

<i>Running Hour</i>	<i>Operation Category</i>	<i>Fuel Consumption (L)</i>
01:00	Full Speed	28
02:30	Economical Speed	45
00:30	Slow Speed/Manuever	5.5
00:10	Idle Speed	1
Total		79.5

Metode perhitungan bahan bakar ini nantinya akan diimplementasi pada satu halaman spesifik yang menampilkan konsumsi bahan bakar dalam rentang satu hari. Di halaman ini, pengguna dapat melakukan filter tanggal untuk memudahkan pemantauan data secara historis. Selain pihak manajemen, awak kapal juga dapat mengakses halaman ini sebagai acuan dalam pengisian nilai *running hour* yang sudah secara otomatis dikategorikan oleh sistem. Halaman ini merupakan salah satu fitur inti dari Sistem Monitoring yang akan dibuat dalam melakukan pemantauan bahan bakar berdasarkan perhitungan yang telah disepakati.

2.15 Penelitian Terdahulu

Berikut rangkuman hasil penelitian terdahulu yang memiliki keterkaitan dengan penelitian yang dilakukan.

Tabel 2.5. Penelitian terdahulu mengenai *Internet of Things* (IoT)

No	Nama Peneliti dan Tahun	Penelitian yang dilakukan
1	Abdulmalek dkk. (2022)	<p>Judul: <i>IoT-Based Healthcare-Monitoring System towards Improving Quality of Life: A Review</i></p> <p>Permasalahan:</p> <p>Kelemahan utama dari layanan kesehatan adalah hanya tersedia di rumah sakit, sehingga tidak memadai dan terkadang tidak mampu memenuhi kebutuhan lansia dan penyandang disabilitas. Pemantauan status kesehatan lansia secara <i>real-time</i> adalah masalah yang diselesaikan secara efektif dan praktis oleh <i>Internet of Things</i> (IoT) dengan penggunaan data sensor dan telekomunikasi. Hasil: Sistem kesehatan berbasis IoT memfasilitasi hidup orang dalam banyak cara.</p>

No	Nama Peneliti dan Tahun	Penelitian yang dilakukan
----	-------------------------	---------------------------

1. ***Remote healthcare:*** Daripada pasien mendatangi layanan kesehatan, solusi nirkabel berbasis IoT menghadirkan layanan kesehatan kepada pasien. Sensor berbasis IoT digunakan untuk mengumpulkan data dengan aman, yang kemudian diproses oleh algoritma kecil dan dibagikan kepada penyedia layanan kesehatan untuk mendapatkan rekomendasi yang tepat.
 2. ***Realtime monitoring:*** Sensor pemantauan berbasis IoT mengumpulkan serangkaian data psikologis. Penyimpanan data dikelola melalui analisis dan *gateway* berbasis *cloud*.
 3. ***Preventive care:*** Data sensor digunakan oleh sistem layanan kesehatan IoT untuk memberi tahu anggota keluarga dan membantu deteksi dini keadaan darurat. *Internet of Things* memungkinkan machine learning untuk deteksi anomali dini dan pelacakan tren kesehatan.
-

No	Nama Peneliti dan Tahun	Penelitian yang dilakukan
2	Anh dkk. (n.d)	<p>Judul: <i>Development and Implementation of a low-cost IoT System for Small Farm Households</i></p> <p>Permasalahan: Pertanian kecil memiliki peran yang penting untuk produksi agrikultural terutama pada negara kurang berkembang maupun berkembang. Berbeda dengan pertanian skala besar yang berinvestasi pada teknologi mutakhir untuk memastikan kualitas hasil panen yang maksimal, teknologi pada pertanian kecil masih sangat terbatas.</p> <p>Hasil: Sistem IoT diusulkan untuk dapat membantu petani kecil meningkatkan kualitas produk pertanian sekaligus mengurangi biaya produksi dan mencegah pemborosan air irigasi dan pupuk. Agrikultur sangat bergantung pada cuaca dan iklim, seperti temperatur dan kadar air tanah. Dalam penelitian, sistem melakukan monitoring pada temperatur, kelembapan, intensitas cahaya, dan kadar air tanah. Parameter tersebut digunakan sebagai acuan dalam mengatur pompa embun, pompa irigasi, jendela ventilasi, kipas ventilasi, dan grow light.</p>
3	Hizbullah, Djohar, dan Mabud (2022)	<p>Judul: <i>Internet of Things for Smart Transportation in North Moluccas Province</i></p> <p>Permasalahan: Perlunya transportasi yang lebih aman dan penyediaan layanan keselamatan selama keadaan darurat di wilayah Provinsi Maluku Utara.</p>

No	Nama Peneliti dan Tahun	Penelitian yang dilakukan
		<p>Hasil: Diterapkan otomasi pada sistem navigasi yang dapat membantu meningkatkan akurasi dan keandalan navigasi perahu agar mengurangi risiko kecelakaan. Peneliti juga menerapkan sistem monitoring yang dapat menyediakan data secara <i>real-time</i> terhadap kondisi perahu untuk kebutuhan <i>maintenance</i> dan deteksi lebih awal isu yang mungkin akan terjadi.</p>
4	Maswadi, Ghani, dan Hamid (2020)	<p>Judul: <i>Systematic Literature Review of Smart Home Monitoring Technologies Based on IoT for the Elderly</i></p> <p>Permasalahan: Dengan seiring bertambahnya populasi lansia berumur 65 keatas di negara-negara seperti Amerika, Jerman, Perancis, Itali, dan Jepang terdapat kemungkinan mereka akan beban yang bertambah pada kesehatan dan layanan sosial. Diperlukan teknologi yang dapat memberikan lingkungan hidup yang kondusif bagi para lansia.</p> <p>Hasil: Penerapan teknologi sistem smart home pada lansia telah secara signifikan meningkatkan kualitas hidup diantara para lansia. Beberapa teknologi yang dilaporkan telah menyelamatkan hidup para lansia di situasi darurat.</p>

No	Nama Peneliti dan Tahun	Penelitian yang dilakukan
5	Song dkk. (2022)	<p>Judul: <i>Internet of Maritime Things Platform for Remote Marine Water Quality Monitoring</i></p> <p>Permasalahan: Penerapan sistem monitoring kualitas air di laut memerlukan dukungan komunikasi jarak jauh dan berkecepatan tinggi yang stabil.</p> <p>Hasil: Dalam penelitian ini, dikembangkan sebuah platform IoT Maritim yang mendukung komunikasi jarak jauh dan berkecepatan tinggi untuk pemantauan kualitas air laut jarak jauh dan online. Perangkat ditempatkan di atas permukaan air laut dan gerbang untuk pengiriman data ditempatkan darat. Untuk merealisasi komunikasi jarak jauh dan berkecepatan tinggi antara perangkat dengan control center di darat, dikembangkan sistem penyesuaian sinar otomatis (<i>automatic beam adjustment system</i>) untuk antena pengarah sehingga dapat mendukung komunikasi jarak jauh dan berkecepatan tinggi dengan secara otomatis mengatur derajat sinar agar selalu mengarah ke <i>gateway</i> di darat. Metode ini terbukti memberikan performa komunikasi dua kali lipat dibandingkan koneksi nirkabel (LTE di laut) yang ada.</p>

BAB III

METODE PENELITIAN

3.1 Garis Besar Penelitian

Secara garis besar, penelitian ini akan melaksanakan pengembangan sistem monitoring mesin diesel berbasis *IoT* yang akan diuji pada PT Bisma Jaya untuk meningkatkan efektivitas operasi dengan menekankan kecepatan minimum pada armada dan efisiensi biaya bahan bakar dengan memantau jumlah bahan bakar yang digunakan setiap harinya. Tampilan web akan dikembangkan menggunakan *framework* NextJS berbasis JavaScript dan sistem *backend* menggunakan *framework* Django berbasis Python. Pengembangan sistem ini akan menggunakan metode *Extreme Programming* yang memiliki tahapan sebagai berikut: *Exploration Phase*, *Planning Phase*, *Iteration to Release Phase*, *Productionizing Phase*, *Maintenance Phase*, dan *Death Phase*.

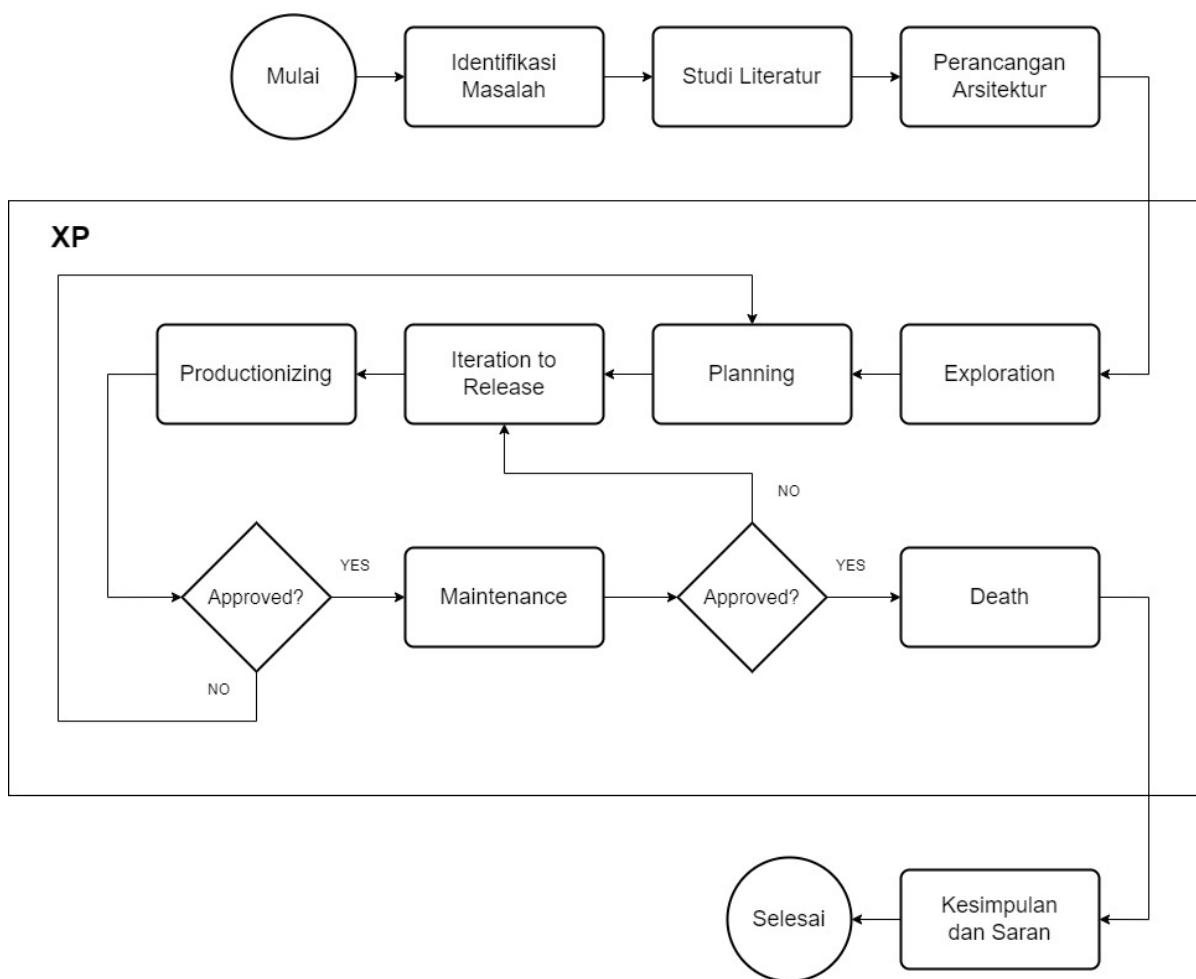
3.2 Diagram Alir Penelitian

Metodologi pelaksanaan penelitian ini dapat dimodelkan menggunakan diagram alir sebagai berikut.

Pada Gambar 3.1 dapat dilihat bahwa penelitian ini diawali dengan melakukan identifikasi masalah dan studi literatur mengenai jurnal-jurnal dari penelitian relevan yang telah dilakukan sebelumnya. Kemudian, dilanjutkan dengan pengembangan sistem menggunakan metode *Exteme Programming (XP)* yang terdiri dari *Exploration Phase*, *Planning Phase*, *Iteration to Release Phase*, *Productionizing Phase*, *Maintenance Phase*, dan *Death Phase*. Terakhir dilakukan pembuatan kesimpulan dan saran dari penelitian.

3.3 Prosedur Penelitian

Berikut penjelasan prosedur penelitian secara rinci berdasarkan tahapan-tahapan yang ditetapkan pada Gambar 3.1 sebagai pedoman penelitian ini.



Gambar 3.1. Diagram Alir Penelitian

3.3.1 Identifikasi Masalah

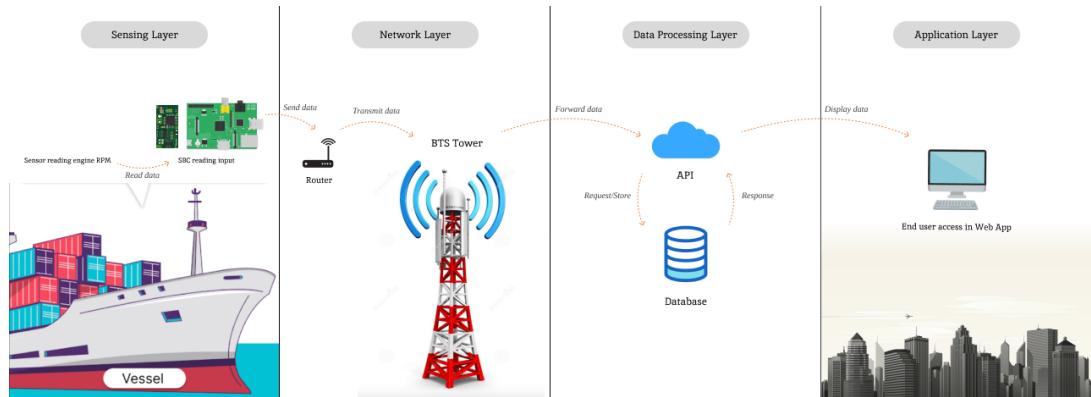
Penelitian ini dimulai dengan mengidentifikasi permasalahan yang ada di mitra. Ini dilakukan melalui dua cara, diskusi dengan seluruh *stakeholder* yang terlibat dan observasi ke lapangan. Dari tahap ini akan dihasilkan rumusan masalah, tujuan, serta batasan-batasan penelitian.

3.3.2 Studi Literatur

Pada tahap ini dilakukan studi literatur dengan mengumpulkan berbagai referensi yang didapat dari jurnal, artikel ilmiah, dan sumber lainnya mengenai pengembangan sistem berbasis IoT, teknologi-teknologi yang digunakan selama penelitian, dan perbandingan metodologi dalam pengembangan perangkat lunak.

3.3.3 Pengembangan Sistem

3.3.3.1 Perancangan Arsitektur



Gambar 3.2. Arsitektur Sistem Monitoring berbasis IoT

Gambar diatas merupakan arsitektur dari Sistem Monitoring berbasis IoT yang akan dikembangkan. Pertama, perangkat IoT akan dipasang di kapal untuk mengumpulkan data dari sensor. Data ini kemudian akan dikirim oleh *router/modem LTE* menuju server melalui API yang telah dibuat dan disimpan ke dalam basis data. Sebelum ditampilkan di aplikasi web, data ini akan diproses terlebih dahulu di *Data Processing Layer* sehingga menghasilkan wawasan seperti data *running hour* mesin, tren data kecepatan mesin, dan tren data konsumsi bahan bakar. Di lapisan terakhir, data akan ditampilkan dalam bentuk grafik untuk menganalisis tren data kecepatan dan konsumsi bahan bakar serta log data untuk mendapatkan informasi spesifik pada rentang waktu tertentu. Pada penelitian ini, difokuskan pada pengembangan untuk *Network Layer*, *Data Processing Layer*, dan *App Layer* berdasarkan arsitektur IoT yang dibahas pada bab sebelumnya.

3.3.3.2 Extreme Programming

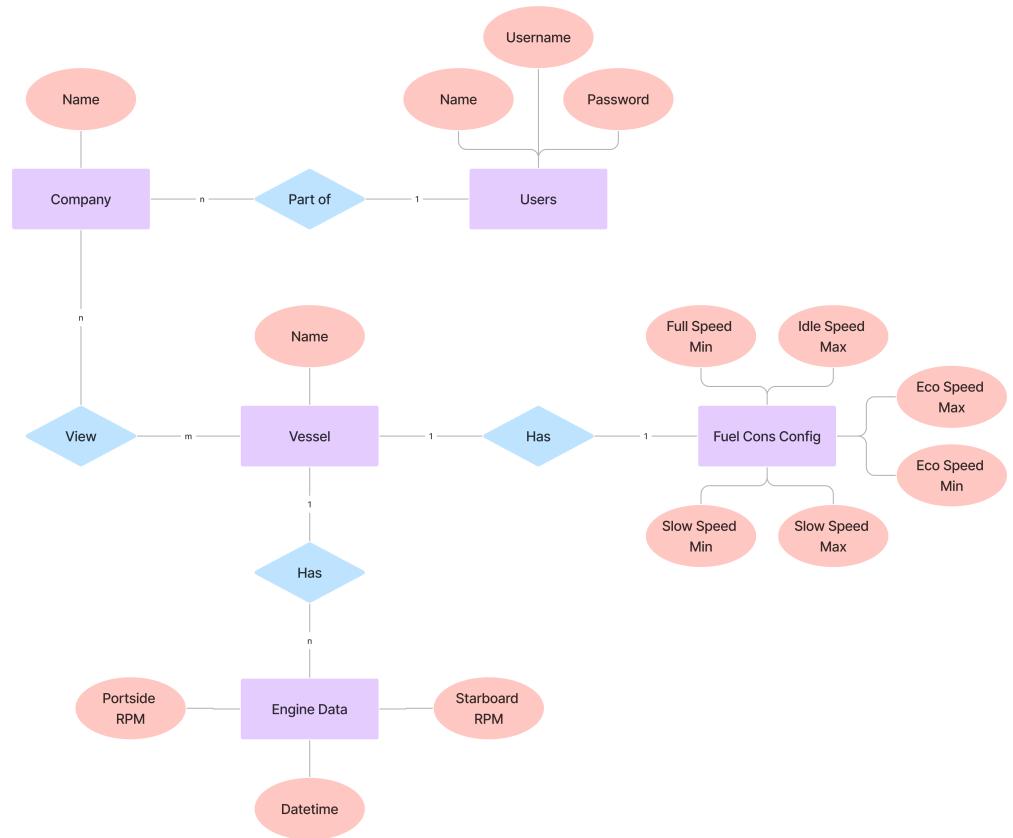
Exploration Phase Kebutuhan yang dikumpulkan pada tahap identifikasi masalah kemudian dibuat dalam bentuk *user story* untuk mendeskripsikan hasil yang diinginkan. Daftar *user story* sementara dapat dilihat pada tabel berikut.

Tabel 3.1. Daftar *user story* rencana awal

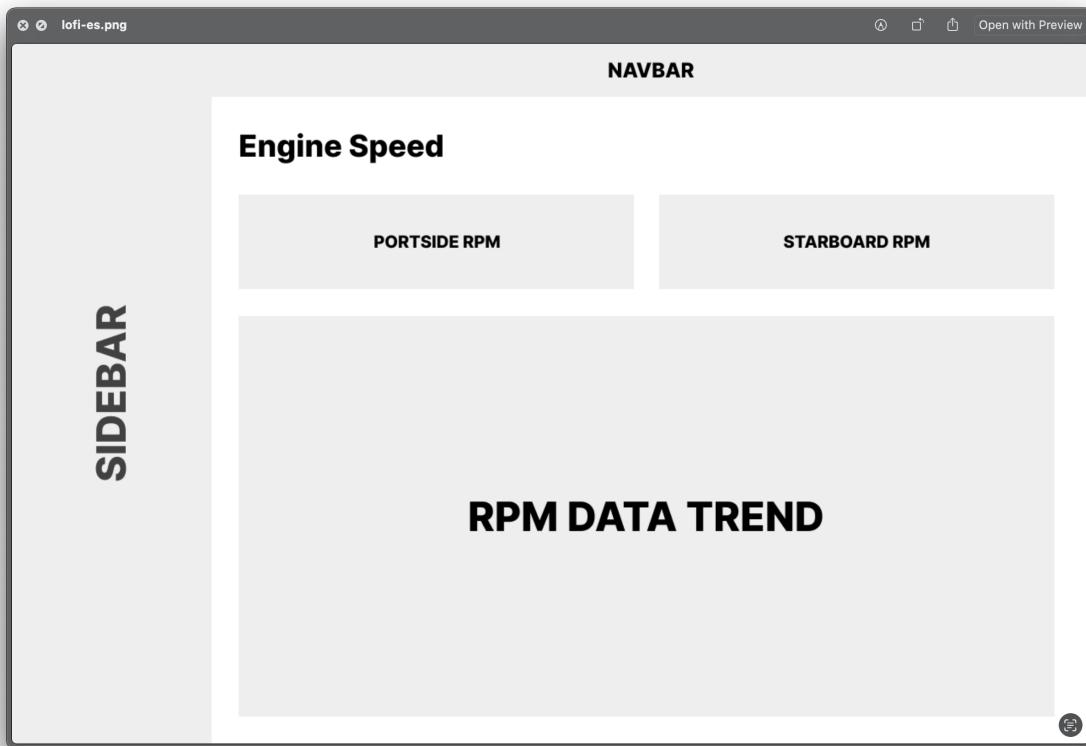
<i>Code</i>	<i>Persona</i>	<i>I want to</i>	<i>So that can</i>
US-01	<i>User</i>	<i>Login</i>	Mengakses sistem sesuai dengan <i>username</i> dan <i>password</i>
US-02	<i>User</i>	<i>Logout</i>	Keluar dari sistem melalui tombol <i>logout</i>
US-03	<i>User</i>	Melihat data historis kecepatan mesin	Memastikan armada bergerak dengan kecepatan mesin yang sesuai
US-04	<i>User</i>	Melihat data historis konsumsi bahan bakar	Melakukan kontrol bahan bakar
US-05	<i>User</i>	Melihat data historis <i>running hour</i>	Memastikan operasi berjalan dengan optimal
US-06	<i>User</i>	Melihat data log kecepatan per menit	Melihat detail kecepatan pada waktu spesifik
US-07	<i>User</i>	Mencetak laporan kecepatan mesin harian	Mendapatkan laporan harian kecepatan mesin dengan format PDF
US-08	<i>User</i>	Mencetak laporan harian konsumsi bahan bakar	Mendapatkan laporan harian konsumsi bahan bakar dengan format PDF
US-09	<i>User</i>	Mengunduh data log kecepatan mesin	Mendapatkan log kecepatan mesin dengan format CSV

Lebih lanjut, berdasarkan *user story* yang dibuat di tahap sebelumnya, akan dilakukan analisis dengan output perancangan tampilan dalam bentuk *wireframe* dan skema basis data dalam bentuk *entity relationship diagram* (ERD) yang akan membantu dalam menggambarkan hubungan antar tabel. ERD dan contoh *wireframe*

dapat dilihat pada Gambar 3.3 dan Gambar 3.4 secara berturut-turut.



Gambar 3.3. *Entity Relationship Diagram (ERD)* Sementara



Gambar 3.4. *Wireframe* Halaman Engine Speed

Planning Phase *User story* yang dibuat pada tahap sebelumnya akan dikumpulkan dan disimpan berdasarkan prioritas dan tingkat kesulitan/waktu pengerjaannya. Tahap ini akan diulang kembali setelah melewati tahap pengujian sesuai dengan jumlah iterasi pengembangan.

Iteration to Release Phase Fase ini terbagi menjadi beberapa bagian. Pertama, dilakukan analisis kebutuhan yang akan menghasilkan kebutuhan fungsional sistem dari task yang ditentukan; Kemudian, dilakukan desain *wireframe* sederhana untuk membantu pengembang dalam proses penyelesaian *task* dengan sesuai; Lalu, dilakukan tahap implementasi dari perancangan sistem. Dalam pengembangan dasbor digunakan *framework* NextJS untuk menghasilkan tampilan yang dinamis, kemudian digunakan *framework* Django yang berbasis bahasa Python untuk mengembangkan API agar perangkat IoT dapat mengirim data ke server. Setelah dilakukannya pengembangan seluruh *task* pada iterasi tersebut, setiap fungsi akan diuji melalui

unit test sebelum akhirnya diunggah ke repositori dan akan diuji kembali setelahnya oleh mitra melalui metode *black box testing*. Jika terdapat umpan balik, maka tahapan akan diulang ke tahap desain untuk menyesuaikan permintaan mitra. Sebaliknya, jika mendapatkan persetujuan maka sistem akan masuk ke tahap berikutnya.

Productionizing Phase Setelah melewati serangkaian pengujian dan mendapatkan persetujuan mitra, sistem akan diluncurkan secara perlahan ke *server* dengan mode *development*. Pada fase ini, mitra dapat langsung menggunakan sistem yang belum sempurna dengan harapan dapat memberi umpan balik selama proses pengembangan sistem. Jika terdapat fitur yang dirasa kurang lengkap atau ingin ditambahkan oleh mitra, hasilnya akan dibalikkan ke tahap *planning* untuk dilakukan prioritas ulang.

Maintenance Phase Pada tahap ini, sistem sudah hampir matang dan menunggu persetujuan terakhir dari mitra dengan mengisi *User Acceptance Test* (UAT). Jika ditemukan *bug* atau ketidak sesuaian pada sistem maka akan kembali ke tahap *Iteration to Release* untuk dilakukan *debugging*, sebaliknya jika hasil UAT dari mitra menunjukkan sistem sudah layak dioperasikan secara penuh maka akan lanjut ke tahap terakhir.

Death Phase Ini merupakan tahap terakhir, apabila seluruh fitur pada sistem telah selesai dikembangkan dan sesuai dengan kebutuhan awal mitra maka dilakukan tahap peluncuran dimana akan digunakan *VPS cloud hosting* sebagai tempat sistem diluncurkan.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Perancangan (*Exploration Phase*)

4.1.1 Requirements

Pada fase ini, seluruh kebutuhan pengguna akan dikumpulkan dalam bentuk *user story*. Seluruh *user story* akan diubah menjadi *task* yang dapat dikerjakan pada jangka waktu tertentu yang disebut dengan iterasi. Pada metode *Extreme Programming*, iterasi dapat berlangsung selama 1 hingga 2 pekan. Dalam penelitian ini, 1 iterasi akan ditetapkan berlangsung selama 1 pekan. *User story* pada penelitian ini dapat dilihat pada tabel dibawah.

Tabel 4.1. Daftar *user story* Sistem Monitoring

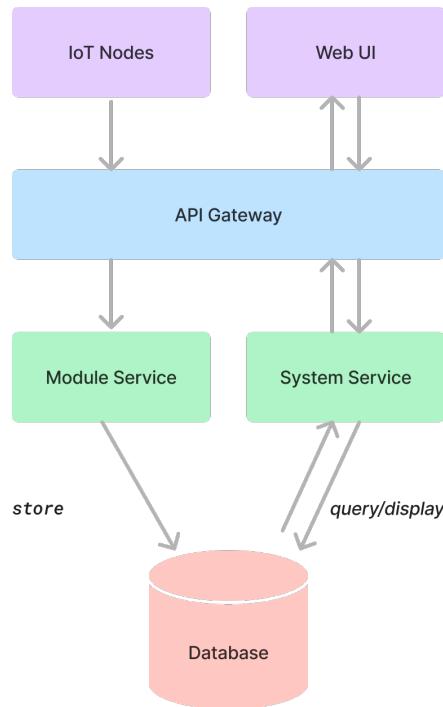
<i>Code</i>	<i>Persona</i>	<i>I want to</i>	<i>So that can</i>
US-01	User	Login	Mengakses sistem sesuai dengan username dan password
US-02	User	Logout	Keluar dari sistem melalui tombol logout
US-03	User	Melihat data historis kecepatan mesin	Memastikan armada bergerak dengan kecepatan mesin yang sesuai
US-04	User	Melihat data historis konsumsi bahan bakar	Melakukan kontrol bahan bakar
US-05	User	Melihat data historis running hour	Memastikan operasi berjalan dengan optimal
US-06	User	Melihat data log kecepatan mesin dalam interval 1 menit	Melihat detail ekcepatan pada waktu spesifik
US-07	User	Mencetak laporan kecepatan mesin harian	Mendapatkan laporan harian kecepatan mesin dengan format PDF

Code	Persona	I want to	So that can
US-08	User	Mencetak laporan harian konsumsi bahan bakar	Mendapatkan laporan harian konsumsi bahan bakar dengan format PDF
US-09	User	Mengunduh data log kecepatan mesin	Mendapatkan log kecepatan mesin dengan format CSV
US-10	Admin	Login	Mengakses sistem sesuai dengan username dan password
US-11	Admin	Logout	Keluar dari sistem melalui tombol logout
US-12	Admin	Mengelola data pengguna	Mengelola data pengguna yang dapat mengakses sistem
US-13	Admin	Mengelola data kapals	Mengelola data profil kapal
US-14	Admin	Mengelola data FCRV	Menetapkan nilai batas atas/bawah tiap kategori kecepatan
US-15	User	Melihat informasi konsumsi bahan bakar dan running hour kapal pada periode tertentu	Mendapat ringkasan informasi dari kapal
US-16	User	Melihat data konsumsi bahan bakar dalam format dokumen OP41	Mengarahkan awak kapal untuk mengacu nilai laporan dari halaman tersebut
US-17	User	Memilih kapal pada satu halaman tertentu	Melakukan navigasi ke halaman detail kapal tertentu

4.1.2 Architecture Modelling

Dalam buku *Fundamental of Software Architecture* oleh Richards dan Ford (2020), arsitektur didefinisikan sebagai sebuah rangkaian dari keputusan yang akan

menentukan struktur dan perilaku sistem. Pada metode XP, model arsitektur dibuat untuk mempertimbangkan berbagai solusi alternatif. Model arsitektur pada Sistem Monitoring dapat dilihat pada Gambar 4.1 .



Gambar 4.1. Metafor Model Arsitektur Sistem Monitoring

Gambar 4.1 merupakan model arsitektur dengan style *Service-based Architecture* dengan pertimbangan bahwa sistem tidak hanya menyediakan (*API Gateway*) untuk web dasbor saja, melainkan juga untuk perangkat IoT yang akan dipasang di kapal, disebut juga dengan *IoT Nodes*. *IoT Nodes* dapat mengirimkan data ke server melalui (*API Gateway*) yang telah diatur pada *Module Service* untuk meneruskan data ke *Database*. Data yang telah tersimpan dapat diakses melalui *Web UI* yang terhubung dengan *System Service* melalui (*API Gateway*). Selain mengakses data, sistem juga memungkinkan pengguna untuk dapat melakukan filter data dengan memberi input tanggal yang ditunjukkan oleh dua garis penghubung pada setiap obyek yang menghubungkan *Web UI* dengan *Database*.

4.1.3 Tools dan Teknologi

Selama pengembangan sistem, digunakan beberapa *tools* yang akan membantu proses tersebut serta pemilihan teknologi (*techstack*) untuk Sistem Monitoring yang sebelumnya telah dibahas pada Bab 2 dan Bab 3. Untuk tools yang digunakan dalam penelitian ini meliputi VSCode sebagai IDE, phpmyadmin untuk melakukan manajemen basis data, Figma untuk mendesain arsitektur, skema basis data, dan prototype, serta Apidog untuk pengujian API.

4.2 Perencanaan (*Planning*)

Pada tahap ini, seluruh *User Story* akan menjadi kumpulan *task* yang akan disortir berdasarkan tingkat prioritas yang dinilai dari 1 hingga 5 dan estimasi waktu pengerjaannya dalam satuan hari. Seluruh rencana iterasi dapat dilihat pada Tabel 4.2 hingga Tabel 4.6 dibawah.

Iterasi pertama difokuskan untuk mengelola data kecepatan mesin dan menampilkannya dalam bentuk grafik.

Tabel 4.2. Rencana Iterasi 1

Kode User Story	Deskripsi Task	Prioritas	Estimasi Waktu
US-03	Mendapatkan informasi data historis rata-rata kecepatan mesin dalam interval 1 jam selama 1 hari	5	4

Pada iterasi kedua, dilanjutkan pengembangan Halaman Fuel Consumption, Running Hour, dan Data Log.

Tabel 4.3. Rencana Iterasi 2

Kode Story	User	Deskripsi Task	Prioritas	Estimasi Waktu
US-04		Mendapatkan informasi data historis rata-rata konsumsi bahan bakar dalam interval 1 jam selama 1 hari	5	3
US-05		Mendapatkan informasi data historis running hour setiap tanggal	4	2
US-06		Mendapatkan informasi data log kecepatan mesin dalam interval 1 menit	4	1

Selanjutnya, pada iterasi ketiga dilakukan pengembangan sistem admin yang memungkinkan pengguna melakukan manajemen data seperti data pengguna, data kapal, dan data batas kecepatan pada setiap kategori operasional FCRV.

Tabel 4.4. Rencana Iterasi 3

Kode Story	User	Deskripsi Task	Prioritas	Estimasi Waktu
US-14		Admin melakukan manajemen nilai batas atas/bawah keceptan mesin sesuai kategori operasi FCRV	4	1
US-12		Admin melakukan manajemen data pengguna	3	1
US-13		Admin melakukan manajemen data kapal	3	1

Kode	User	Deskripsi	Prioritas	Estimasi
Story			Waktu	
US-10		Admin melakukan login sebelum masuk ke sistem admin	2	1
US-11		Admin keluar dari sistem admin melalui tombol logout	2	1

Iterasi keempat berfokus pada halaman Overview, ekspor data mentah kecepatan mesin dalam format CSV, dan laporan harian kecepatan mesin dan konsumsi bahan bakar dalam format PDF

Tabel 4.5. Rencana Iterasi 4

Kode	User	Deskripsi Task	Prioritas	Estimasi
Story			Waktu	
US-15		Mendapatkan ringkasan informasi konsumsi bahan bakar dan running hour kapal	3	1
US-07		Mengunduh laporan harian kecepatan mesin dengan format PDF	2	1
US-08		Mengunduh laporan harian konsumsi bahan bakar dengan format PDF	2	1
US-09		Mengunduh seluruh log data dari modul dengan format CSV	2	1

Terakhir, pada iterasi kelima dilakukan pengembangan halaman OP41 Report, halaman Home, dan sistem autentikasi.

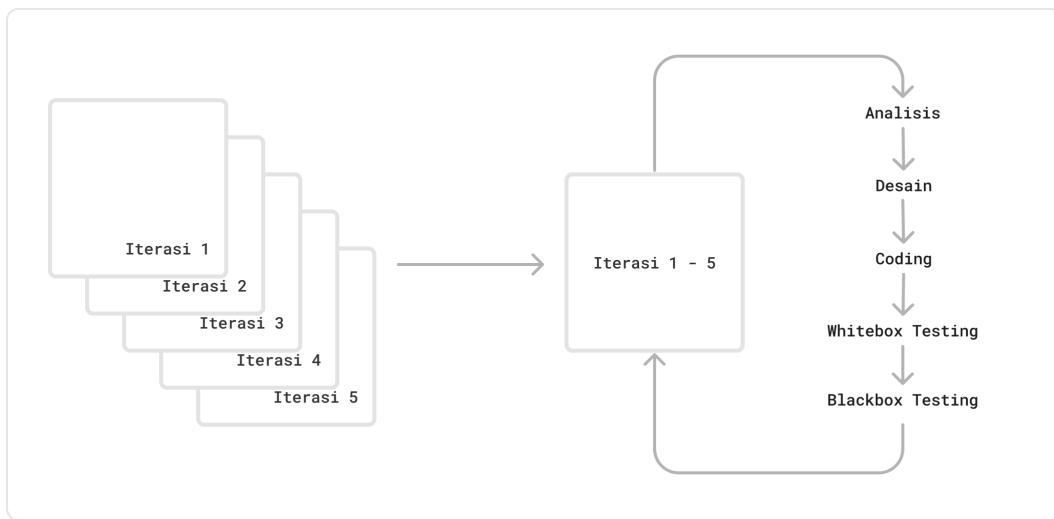
Tabel 4.6. Rencana Iterasi 5

Kode Story	User	Deskripsi Task	Prioritas	Estimasi Waktu
US-16		Menampilkan data konsumsi bahan bakar dan running hour yang ditampilkan berdasarkan kategori operasi dari Dokumen FCRV	2	2
US-17		Menampilkan seluruh data kapal yang terdaftar	2	1
US-01		Sistem login agar hanya pengguna dengan terautentikasi yang dapat mengakses sistem	2	1
US-02		Keluar dari sistem melalui tombol logout	2	1

4.3 Implementasi (*Iteration to Release*)

Secara garis besar, implementasi dilakukan sebanyak 5 iterasi yang berlangsung selama 5 pekan seperti yang terlihat pada Gambar 4.2. Dimulai dari tahap analisis yang akan menghasilkan kebutuhan sistem, dilanjutkan dengan perancangan *wireframe* sebagai acuan pada pengembangan antarmuka di fase koding, yang akan melewati serangkaian proses pengujian yang disebut dengan *Whitebox Testing* dan *Blackbox Testing*. Pengujian *whitebox* dilakukan untuk memastikan kesesuaian seluruh struktur dan logika pemrograman dengan melalui *unit test*. Lalu, setelahnya dilakukan pengujian langsung oleh mitra untuk memastikan input dan output sistem telah sesuai.

Setelah satu iterasi selesai dan telah melewati seluruh ceklis pada *Whitebox* dan *Blackbox testing* maka akan dilanjutkan ke iterasi kedua. Pada penelitian ini, hasil iterasi 2 hingga iterasi 5 ditaruh pada lampiran dikarenakan memiliki pola



Gambar 4.2. Garis Besar Pengerjaan Implementasi

pengembangan yang identikal. Hasil pada iterasi 1 akan secara detail dijabarkan pada bagian-bagian berikut.

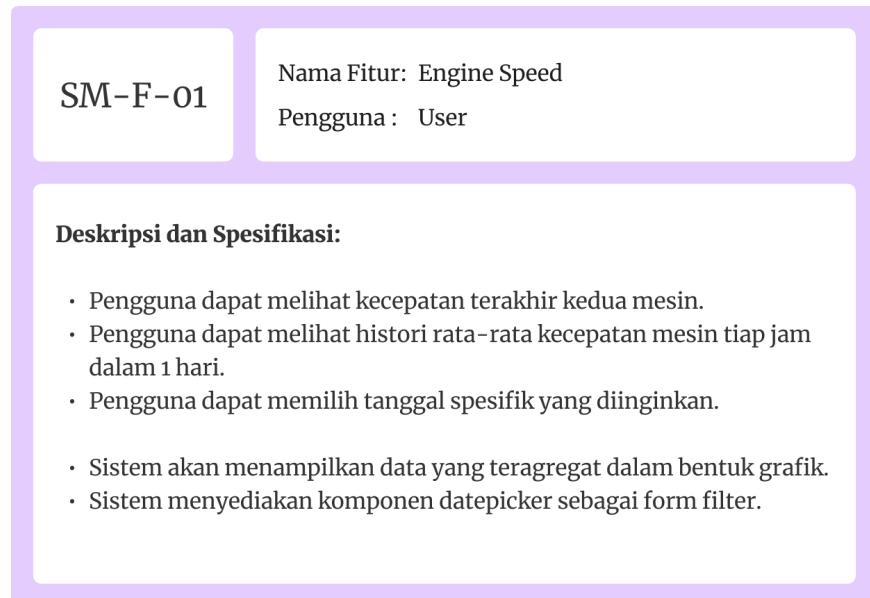
4.3.1 *Analysis*

Kebutuhan sistem merupakan analisis yang dilakukan untuk mengetahui kebutuhan fungsional sistem. Kebutuhan fungsional sistem sendiri merupakan fungsionalitas yang harus tersedia di sistem sesuai kebutuhan *stakeholder* yang tertuang di *user story*. Aturan penomoran kebutuhan sistem dapat dilihat pada Tabel 4.7 .

Tabel 4.7. Aturan Penomoran Kebutuhan Sistem

Kode	Keterangan
SM-	Sistem Monitoring
-F-	Kebutuhan Fungsional
-{x}	Nomor Urutan Kode Kebutuhan

Berikut merupakan contoh kebutuhan fungsional pada iterasi 1 yang hanya meliputi fitur Engine Speed saja dikarenakan estimasi waktu pengerjaan yang memakan waktu relatif lebih lama. Kebutuhan fungsional memuat setidaknya kode fungsional, nama fitur, pengguna, serta deskripsi dan spesifikasi.

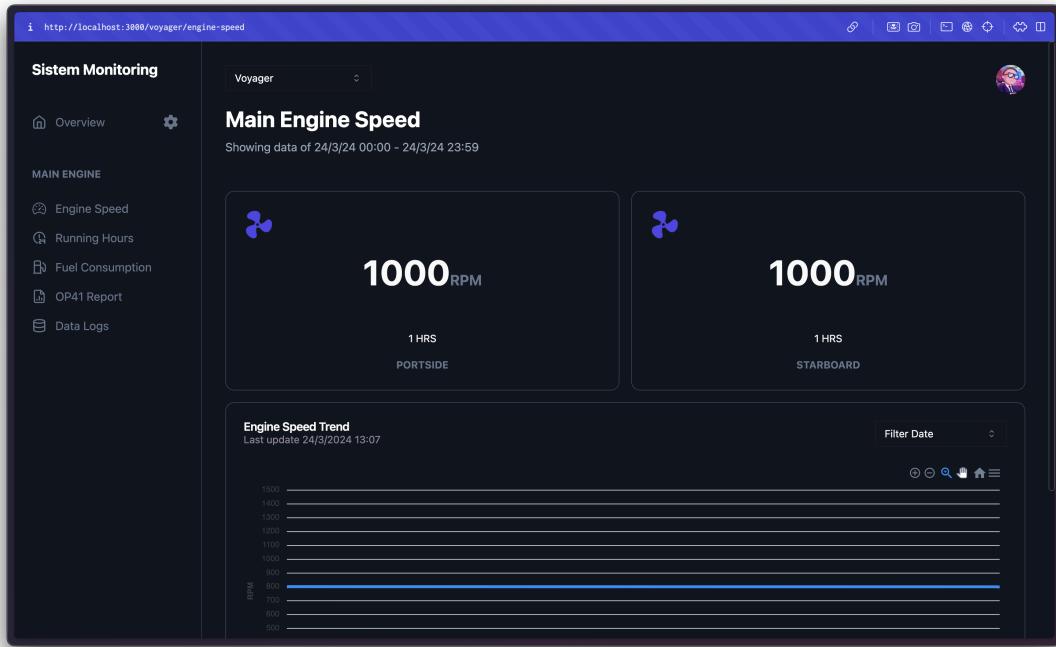


Gambar 4.3. Kebutuhan Fungsional Engine Speed

4.3.2 *Design*

Lalu, pada tahap desain dilakukan perancangan *wireframe low fidelity* untuk memberikan gambaran pada pengembang terkait antarmuka dari *task* yang dikerjakan. *Wireframe* terdiri dari komponen dan tata letaknya tanpa teks atau konten detail. *Wireframe* pada Halaman Engine Speed sendiri sebelumnya telah ditampilkan pada Gambar 3.4.

Dari *wireframe* tersebut, dilakukan pengerjaan tampilan yang disebut juga dengan *UI Slicing*. Pada halaman ini, pengguna dapat melihat angka kecepatan mesin terakhir dan nilai rata-rata dengan interval 1 jam dalam jangka waktu satu hari yang disajikan dalam bentuk grafik. Jangka waktu dapat dipilih melalui filter yang akan disediakan pada area komponen grafik. Seluruh navigasi pada sistem dapat dilakukan dengan memilih komponen *Navbar* atau *Sidebar*. Hasil pengerjaan *UI Slicing* dapat dilihat pada Gambar 4.4.



Gambar 4.4. Frontend Halaman Engine Speed

4.3.3 Coding

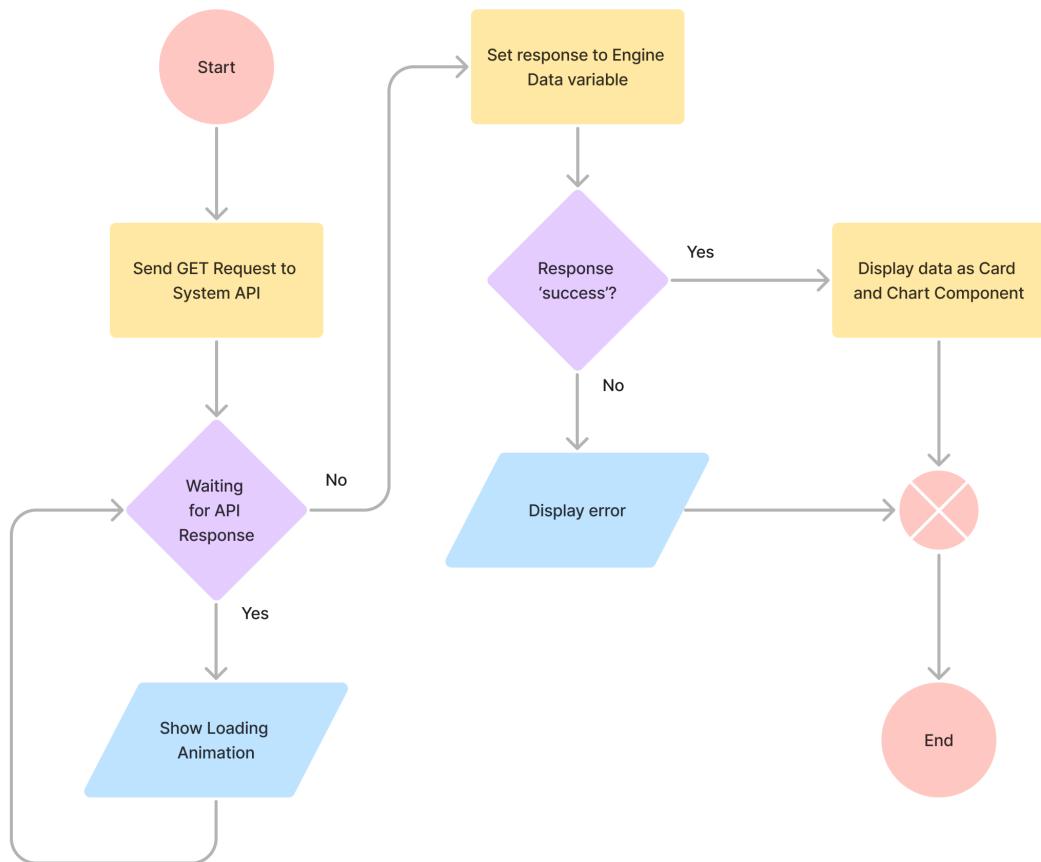
Pada tahap ini, akan dilakukan pengkodean Halaman Engine Speed yang merupakan halaman untuk melihat kecepatan mesin terakhir dan histori tren kecepatan mesin yang disajikan dalam bentuk grafik garis. Algoritma dituangkan dalam bentuk *pseudocode* dan diagram alir yang akan dijabarkan sebagai berikut.

Pada Halaman Engine Speed, data diperoleh dari *server* melalui API berdasarkan kode kapal (*vessel code*) dan filter tanggal. Selama proses pemanggilan data, sistem akan menampilkan animasi *loading* untuk menunjukkan bahwa proses sedang berlangsung. Terdapat dua kemungkinan nilai yang dikembalikan oleh server, yakni '*success*' atau '*error*' beserta dengan penjelasan lengkapnya. Jika '*success*' maka hasil respon dari API akan diinisiasi pada variabel *engine data* dan akan ditampilkan pada komponen *card* dan *chart*. Sebaliknya, layar eror akan ditampilkan beserta dengan kode dan pesan erornya. *Pseudocode* dan diagram alir penampilan data di Halaman Engine Speed dapat dilihat pada Tabel 4.8 dan Gambar 4.5.

Tabel 4.8. Pseudocode Halaman Engine Speed

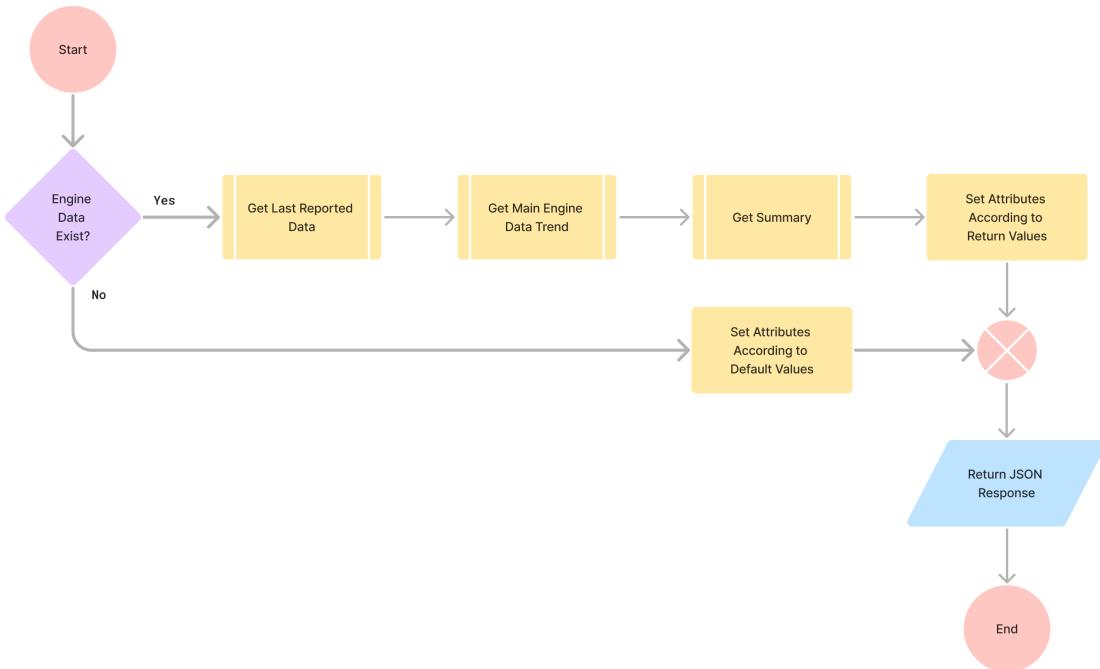
Pseudocode

```
1: GETINITIALDATA(vesselcode, datefilter)
2: response ← GETINITIALDATA(vesselcode, datefilter)
3: output loading animation
4: if response = 'success' then
5:   data ← response.data
6:   output latest engine data and running hour to card component
7:   output engine data trend to line chart
8: else
9:   output error screen
```



Gambar 4.5. Diagram Alir Halaman Engine Speed

Lebih lanjut, berikut hasil dari penggerjaan Engine Speed dari sisi *backend*. Pada satu *endpoint*, terdapat beberapa fungsi yang dipanggil untuk mendapatkan masing-masing data seperti tanggal data terakhir, tren kecepatan mesin, kecepatan mesin terakhir, dan total running hour. Output dari fungsi-fungsi tersebut bergantung pada data kecepatan mesin yang didapatkan dari kueri. Oleh karenanya, dilakukan kontrol *if statement* untuk mengecek apakah data kecepatan mesin kosong atau tidak. Jika kosong, maka fungsi akan mengembalikan nilai *default* yakni 0. Diagram alir pada *backend* Engine Speed dapat dilihat pada Gambar 4.6.



Gambar 4.6. Diagram Alir API Engine Speed

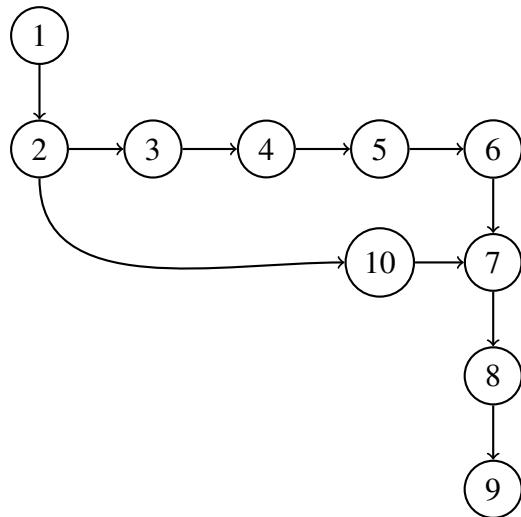
4.3.4 *Whitebox Testing*

Sebelum diunggah ke *codebase* atau Github, dilakukan pengujian *whitebox* dengan membuat *Unit Test* yang akan dijalankan selama pengembangan. *Unit test* dilakukan pada pengembangan *API* yang merupakan bagian dari *Data Processing Layer* dan *Network Layer*. Hal ini agar data yang akan ditampilkan pada antarmuka sistem merupakan data yang valid sehingga dapat memberikan wawasan yang tepat. Aturan penomoran *whitebox* dapat dilihat pada Tabel 4.9.

Tabel 4.9. Aturan Penomoran Whitebox Testing

Kode	Keterangan
SM-	Sistem Monitoring
-WT-	Whitebox Testing
-{x}-	Singkatan Nama Fitur
-{n}	Nomor Urutan Test Case

Hasil perhitungan *cyclomatic complexity* (CYC) pada kode *backend* Engine Speed dimulai dari pembuatan *flowchart* dan *flowgraph* yang dapat dilihat pada Gambar 4.6 dan Gambar 4.7 secara berturut-turut. Lalu dilakukan perhitungan nilai CYC menggunakan rumus yang sebelumnya telah dijabarkan pada Notasi 2.1.



Gambar 4.7. Notasi *Flowgraph Backend Engine Speed*

Dari *flowgraph* diatas, terdapat jumlah Node dan Edge masing-masing adalah 10. Berdasarkan rumus perhitungan 2.1, didapatkan hasil nilai CYC adalah 2. Maka dari itu, telah dibuat 2 *test case* yang dapat dilihat pada Tabel 4.10.

$$\begin{aligned}
 V(G) &= E - N + 2 \\
 &= 10 - 10 + 2 \\
 &= 2
 \end{aligned} \tag{4.1}$$

Namun, pada praktiknya perhitungan nilai CYC tidak dilakukan secara manual, melainkan menggunakan library Python, yakni *mccabe*. Hasil skor CYC pada kode *backend* dapat dilihat pada Lampiran 0.2.1.

Tabel 4.10. Whitebox Testing API Engine Speed

<i>Test Code</i>	<i>Test Case</i>	<i>Scope</i>	<i>Expected Result</i>	<i>Actual Result</i>	<i>Pass/Fail</i>
SM-WT-ES-01	Terdapat data kecepatan mesin di basis data	Engine Speed	Mengembalikan respon sesuai dengan nilai kalkulasi	Sesuai	Pass
SM-WT-ES-02	Tidak terdapat data kecepatan mesin di basis data	Engine Speed	Mengembalikan respon dengan nilai 0	Sesuai	Pass

4.3.5 Blackbox Testing

Setelah repositori telah terbarui, mitra dapat langsung mengakses sistem untuk menguji fungsionalitas fitur yang telah dikerjakan pada iterasi tersebut.

Tabel 4.11. Aturan Penomoran Blackbox Testing

Kode	Keterangan
SM-	Sistem Monitoring
-BT-	Blackbox Testing
-{x}-	Singkatan Nama Fitur
-{n}	Nomor Urutan Test Case

Test case yang dibuat untuk Halaman Engine Speed memastikan grafik telah ditampilkan dengan sesuai dan ketika pengguna melakukan filter tanggal, data grafik akan terubah sesuai dengan data tanggal yang diinput. Berikut hasil *test case* untuk iterasi 1

Tabel 4.12. Blackbox Testing Halaman Engine Speed

Test Code	Test Case	Test Steps	Expected Result	Actual Result	Pass/Fail
SM-BT-ES-01	Melihat data historis harian	1. Memilih kapal yang ingin dilihat 2. Membuka halaman Engine Speed	Sistem menampilkan trend data kecepatan melalui grafik garis pada hari sistem diakses	Sesuai	Pass

<i>Test Code</i>	<i>Test Case</i>	<i>Test Steps</i>	<i>Expected Result</i>	<i>Actual Result</i>	<i>Pass/Fail</i>
SM-BT-ES-02	Melakukan filter tanggal	1. Memilih kapal yang ingin dilihat 2. Membuka halaman Engine Speed 3. Memilih tanggal melalui komponen datepicker	Sistem menampilkan trend data kecepatan melalui line chart pada hari yang dipilih	Sesuai	Pass

4.4 Productionizing Phase

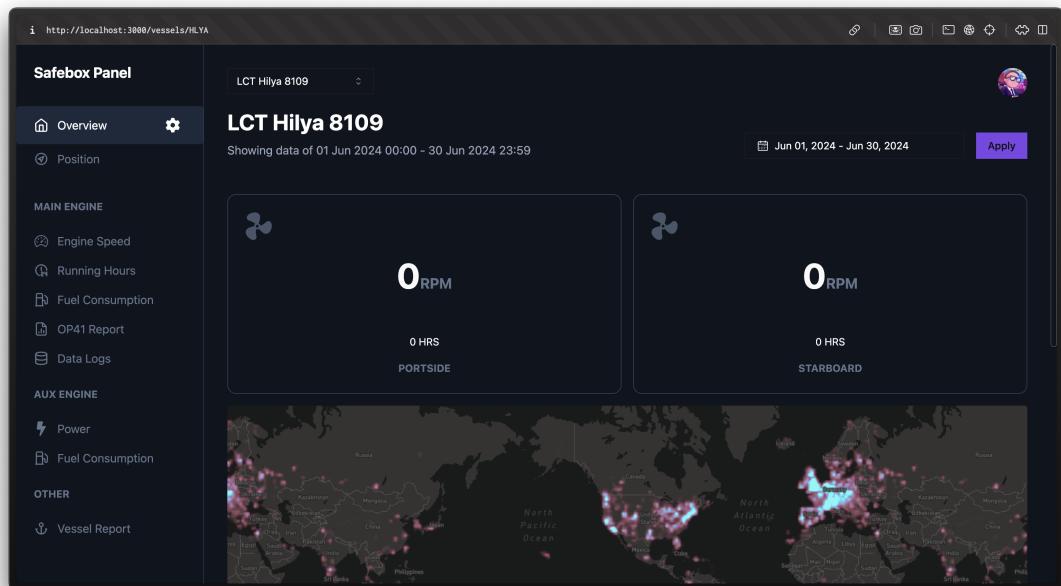
Selama pengerjaan berlangsung, terdapat beberapa umpan balik dari mitra yang mengakibatkan penambahan pada *task*. Sehingga, ini juga berdampak pada rencana iterasi yang sebelumnya telah dibuat. Secara garis besar, *task* yang bertambah adalah sistem admin untuk melakukan manajemen data kapal, manajemen data pengguna, manajemen data FCRV dan beberapa halaman tambahan untuk pengguna. Hasil akhir, umpan balik dapat dilihat pada Tabel 4.13

Tabel 4.13. Umpan Balik Selama Pengembangan

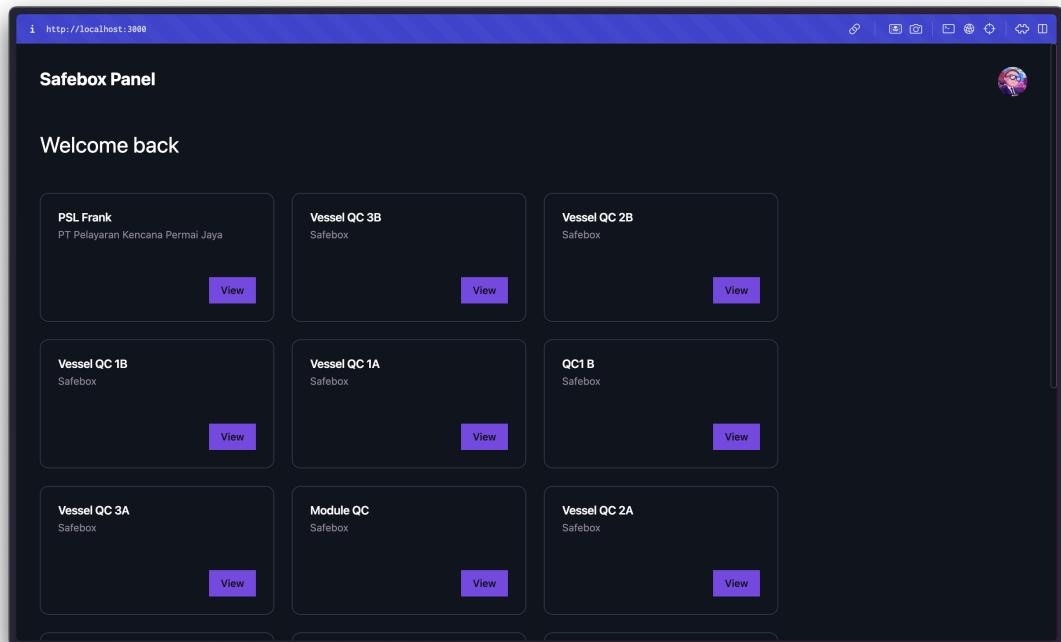
No	Umpam Balik	Iterasi	Status
1	Menambah angka running hour pada halaman engine speed	2	Selesai
2	Menambah sistem admin	3	Selesai
3	Menambah halaman Overview	4	Selesai
4	Menambah halaman Home	5	Selesai
5	Menambah halaman OP41 Report	5	Selesai
6	Menambah peringatan waktu sinkron data pada halaman OP41 Report	5	Selesai

Pada iterasi kedua, terdapat permintaan untuk mencantumkan angka *running hour* pada Halaman Engine Speed yang diposisikan dibawah angka kecepatan mesin. Lalu pada iterasi 3 terdapat permintaan tambahan untuk membangun sistem admin agar mitra dapat lebih leluasa untuk melakukan konfigurasi data. Sistem admin meliputi manajemen data pengguna, data kapal, dan data FCRV. Setelah direkap di *user story* dan dilakukan skala prioritas, pengembangan Sistem Admin akan dilakukan pada iterasi 3. Pada iterasi 4, ditambahkan halaman Overview yang berisi data kecepatan mesin terakhir dan data ringkasan konsumsi bahan bakar yang dapat dilihat pada gambar 4.8. Terakhir, pada iterasi 5 terdapat permintaan untuk menambah 2 halaman, yakni Home dan OP41 Report. Halaman Home berisi seluruh kapal yang terdaftar dan Halaman OP41 Report berisi laporan konsumsi bahan bakar berdasarkan perhitungan

dari kategori FCRV yang dapat dilihat pada Gambar 4.9 dan Gambar 4.10.



Gambar 4.8. Frontend Halaman Overview



Gambar 4.9. Frontend Halaman Home

Category	Running Hour (hh:mm)			Fuel Used (L)		
	Portside	Starboard	Average	Portside	Starboard	Total
Full	00:30	00:30	00:60	29.5	30.5	60
Economical	00:00	00:00	00:00	0	0	0
Slow/Maneuver	00:00	00:00	00:00	0	0	0
Idle	00:30	00:30	00:60	3	3	6
				Total	66	

Note:
Gunakan nilai average ketika menginput angka running hour di laporan
Fuel Calculation based on Fuel Consumption Rate Verification
No. 038/FCRV-FMC/X/2021 by Pertamina Hulu Mahakam

Gambar 4.10. Frontend Halaman OP41 Report

4.5 Maintenance Phase

Pada fase ini, Sistem Monitoring sudah dapat digunakan layaknya produk akhir. Seluruh data yang diinput pada Sistem merupakan data asli yang telah terhubung dengan basis data utama. Fase ini diakhiri dengan penandada tanganan Dokumen User Acceptance Test (UAT) oleh mitra yang dapat dilihat pada Lampiran 0.2.

4.6 Death Phase

Pada akhirnya, konfigurasi lingkungan diubah dari '*development*' menjadi '*production*'. Hasil akhir dari fase ini adalah sebuah sistem yang siap untuk digunakan oleh pengguna.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan penggerjaan tugas akhir yang telah dilakukan, didapatkan kesimpulan sebagai berikut.

1. Sistem Monitoring telah selesai dikembangkan selama 5 iterasi, dimulai dari penggerjaan fitur utama, sistem admin, dan kemudian fitur pendukung dan tambahan. Tiap iterasi memiliki rentang waktu penggerjaan maksimum 1 pekan yang diurutan berdasarkan prioritas dan waktu penggerjaan.
2. Seluruh *task* pada implementasi telah diuji melalui dua metode pengujian, yaitu *Whitebox* dan *Blackbox*. *Whitebox* dilakukan dengan membuat test case pada unit test dan dijalankan sebelum sistem diunggah ke *codebase* dan *Blackbox* dilakukan setelah sistem terunggah ke *codebase* dan masuk ke lingkungan pengembang (*dev mode*).
3. Metode *Extreme Programming* terbukti telah membantu dalam menghadapi perubahan selama pengembangan seperti perubahan pada Iterasi 2, penambahan sistem admin pada Iterasi 3, penambahan halaman Overview, dan penambahan halaman OP41 Report dan Home pada Iterasi 5 tanpa menginterupsi alur dan jadwal pengembangan.

5.2 Saran

Adapun saran untuk penelitian kedepan pada topik ini adalah sebagai berikut.

1. Untuk pengembangan kedepan dapat dilanjut dengan memfokuskan pada keselamatan serta pemantauan emisi pada kapal dengan memasang sensor yang sesuai ataupun dapat memanfaatkan variabel-variabel yang ada.
2. Metode *Extreme Programming* memerlukan disiplin dan komitmen yang tinggi dalam implementasinya, sehingga penggerjaan lebih dari 1 orang

direkomendasikan. Hal ini juga sejalan dengan salah satu praktik metode tersebut, yakni *Pair Programming*.

DAFTAR PUSTAKA

- Abbas, J. (2016). “Quintessence of traditional and agile requirement engineering”. In: *Journal of Software Engineering and Applications* 09 (03), pp. 63–70. doi: 10.4236/jsea.2016.93005.
- Abdulmalek, Suliman dkk. (2022). “IoT-based healthcare-monitoring system towards improving quality of life: A review”. In: *Healthcare*. Vol. 10. 10. MDPI, p. 1993.
- Aditya, Surya (2022). “Main Curang Oknum Kapal Pertamina, 2.520 Liter Solar Dijual Ilegal, Negara Rugi Rp 710 Juta”. news. URL: <https://kaltimkece.id/warta/hukum/main-curang-oknum-kapal-pertamina-2520-liter-solar-dijual-illegal-negara-rugi-rp-710-juta>.
- Anh, Vo Cong dkk. (n.d.). “DEVELOPMENT AND IMPLEMENTATION OF A LOW-COST IOT SYSTEM FOR SMALL FARM HOUSEHOLDS”. In: () .
- Anwer, Faiza dkk. (Mar. 2017). “Comparative Analysis of Two Popular Agile Process Models: Extreme Programming and Scrum”. In: *International Journal of Computer Science and Telecommunications* 8, pp. 1–7.
- Bühler, M., B. Steiner, dan T. Bednar (2022). “Digital twin applications using the simultan data model and python”. In: *Iop Conference Series Earth and Environmental Science* 1101 (8), p. 082015. doi: 10.1088/1755-1315/1101/8/082015.
- Chen, P. (1976). “The entity-relationship model—toward a unified view of data”. In: *Acm Transactions on Database Systems* 1 (1), pp. 9–36. doi: 10.1145/320434.320440.
- Dwitama, Ferliana dan Andre Rusli (2020). “User stories collection via interactive chatbot to support requirements gathering”. In: *TELKOMNIKA (Telecommunication Computing Electronics and Control)* 18.2, pp. 890–898.
- Fahrurrozi, Imam dan SN Azhari (2013). “DENGAN METODE WATERFALL DAN EXTREME PROGRAMMING : STUDI PERBANDINGAN”. In: URL: <https://api.semanticscholar.org/CorpusID:195179186>.
- Gazis, Alexandros (2021). “What is IoT? The Internet of Things explained”. In: URL: <https://api.semanticscholar.org/CorpusID:236342445>.

- Gómez-Hernández, M. dkk. (2023). “Failure to rescue following anatomical lung resection. analysis of a prospective nationwide database”. In: *Frontiers in Surgery* 10. doi: 10.3389/fsurg.2023.1077046.
- Hercog, Darko dkk. (2023). “Design and Implementation of ESP32-Based IoT Devices”. In: *Sensors* 23.15. ISSN: 1424-8220. doi: 10.3390/s23156739. URL: <https://www.mdpi.com/1424-8220/23/15/6739>.
- Hizbullah, Imam, Fahrizal Djohar, dan Zulaeha Mabud (2022). “Internet of Things for Smart Transportation in North Moluccas Province”. In: *MATEC Web of Conferences*. Vol. 372. EDP Sciences, p. 04003.
- Hosny, K. dkk. (2023). “Internet of things applications using raspberry-pi: a survey”. In: *International Journal of Electrical and Computer Engineering (Ijece)* 13 (1), p. 902. doi: 10.11591/ijece.v13i1.pp902-910.
- Johnston, S. dan S. Cox (2017). “The raspberry pi: a technology disrupter, and the enabler of dreams”. In: *Electronics* 6 (3), p. 51. doi: 10.3390/electronics6030051.
- LSE (2023). “What is the blue economy?” URL: <https://www.lse.ac.uk/grantham/institute/explainers/what-is-the-role-of-the-blue-economy-in-a-sustainable-future/>.
- Malloy, B. dan J. Power (2017). “Quantifying the transition from python 2 to 3: an empirical study of python applications”. In: doi: 10.1109/esem.2017.45.
- Maswadi, K., N. Ghani, dan S. Hamid (2020). “Systematic literature review of smart home monitoring technologies based on iot for the elderly”. In: *Ieee Access* 8, pp. 92244–92261. doi: 10.1109/access.2020.2992727.
- Matharu, G. dkk. (2015). “Empirical study of agile software development methodologies”. In: *Acm Sigsoft Software Engineering Notes* 40 (1), pp. 1–6. doi: 10.1145/2693208.2693233.
- Nidhra, Srinivas dan Jagruthi Dondeti (2012). “Black box and white box testing techniques-a literature review”. In: *International Journal of Embedded Systems and Applications (IJESA)* 2.2, pp. 29–50.
- Noerlina, ..., A. F. Chandra, dan T. N. Mursitama (2020). “Sales revenue sharing model using dynamics nav modification in health industries”. In: *IOP Conference Series: Earth and Environmental Science* 426 (1), p. 012162. doi: 10.1088/1755-1315/426/1/012162.

- Perhubungan, Departemen (2020). “Transportasi Laut Pegang Peranan Strategis untuk Merajut Keberagaman Indonesia dan Mendorong Pertumbuhan Ekonomi”. URL: <https://dephub.go.id/post/read/transportasi-laut-pegang-peranan-strategis-untuk-merajut-keberagaman-indonesia-dan-mendorong-pertumbuhan-ekonomi>.
- Raharjana, Indra Kharisma, Daniel Siahaan, dan Chastine Faticahah (2019). “User story extraction from online news for software requirements elicitation: A conceptual model”. In: *2019 16th international joint conference on computer science and software engineering (JCSSE)*. IEEE, pp. 342–347.
- Rahman, Md. Muminur (Jan. 2018). “Implementation of Responsive Online Food Ordering Application with Social Media Integration”. PhD thesis. doi: 10.13140/RG.2.2.32716.36486.
- Rao, G., C. Krishna, dan K. Rao (2013). “Rational unified process for service oriented application in extreme programming”. In: doi: 10.1109/icccnt.2013.6726586.
- Review, British Petroleum Statistical (2016). “Statistical Review of World Energy”. URL: <https://www.bp.com/en/global/corporate/energy-economics/statistical-review-of-world-energy.html>.
- Richards, Mark dan Neal Ford (2020). “Fundamentals of software architecture: an engineering approach”. O'Reilly Media.
- Rumandan, Rhaishudin Jafar (2023). “Development Of Information Systems For Service And Customer Complaints Using The Extreme Programming Method”. In: *Jurnal Teknoinfo* 17.1, pp. 342–350.
- Sikder, Amit Kumar dkk. (Feb. 2018). “A Survey on Sensor-based Threats to Internet-of-Things (IoT) Devices and Applications”. In.
- Song, Yujae dkk. (2022). “Internet of Maritime Things Platform for Remote Marine Water Quality Monitoring”. In: *IEEE Internet of Things Journal* 9.16, pp. 14355–14365. doi: 10.1109/JIOT.2021.3079931.
- Suci, G. dkk. (2023). “Implementation of the arrowhead framework with the beia-iot tool”. In: doi: 10.1117/12.2643257.
- Supriyadi, A., S. Andryana, dan A. Gunaryati (2022). “Perancangan sistem perpustakaan berbasis web”. In: *Jurnal Jtik (Jurnal Teknologi Informasi Dan Komunikasi)* 6 (3), pp. 395–401. doi: 10.35870/jtik.v6i3.439.

- Suryantara, I Gusti Ngurah dan Johanes Fernandes Andry (2018). “Development of Medical Record With Extreme Programming SDLC”. In: *International Journal of New Media Technology*. URL: <https://api.semanticscholar.org/CorpusID:55263555>.
- Tomasdottir, K., M. Aniche, dan A. Deursen (2020). “The adoption of javascript linters in practice: a case study on eslint”. In: *Ieee Transactions on Software Engineering* 46 (8), pp. 863–891. doi: 10.1109/tse.2018.2871058.
- Watson, Arthur Henry, Dolores R Wallace, dan Thomas J McCabe (1996). “Structured testing: A testing methodology using the cyclomatic complexity metric”. Vol. 500. 235. US Department of Commerce, Technology Administration, National Institute of . . .
- Wuryandani, Dewi dan Hilma Meilani (2011). “KEBIJAKAN PENGELOLAAN SUMBER DAYA PERIKANAN LAUT UNTUK MENUNJANG KETAHANAN PANGAN DI INDONESIA”. In: URL: <https://api.semanticscholar.org/CorpusID:168808178>.

LAMPIRAN

0.1 Dokumentasi



Gambar 0.1. Validasi akurasi pembacaan alat oleh PT Bisma Jaya selaku mitra dan PT Pertamina Hulu Mahakam selaku pihak pengguna jasa



Gambar 0.2. Pemasangan dan pengujian alat di lapangan pada salah satu armada yang beroperasi dibawah PT Bisma Jaya

0.2 Hasil Implementasi

Kebutuhan fungsional, *wireframe*, *code snippet*, *unit test*, serta dokumen pendukung dapat dilihat pada tautan berikut: <https://s.itk.ac.id/SMAAppendices>.

0.2.1 Hasil Perhitungan Cyclomatic Complexity

```
❯ radon cc -s views
views/fuel_fcrv.py
  C 9:0 FuelFcrv - A (3)
  M 11:4 FuelFcrv.fuel_consumption - A (2)
views/fuel_fct.py
  C 11:0 FuelFct - A (3)
  M 13:4 FuelFct.fuel_consumption - A (2)
views/fuel_consumption.py
  C 12:0 FuelConsumption - A (2)
  M 15:4 FuelConsumption.get_fuel_per_day - A (1)
views/overview.py
  C 10:0 Overview - A (3)
  M 12:4 Overview.overview - A (2)
views/main_engine.py
  C 9:0 MainEngine - A (3)
  M 11:4 MainEngine.engine_speed - A (2)
  M 46:4 MainEngine.running_hour - A (2)
```

Gambar 0.3. Hasil Perhitungan Cyclomatic Complexity