PERANCANGAN FRONTED APLIKASI PENDUKUNG TEKNOLOGI PERIKANAN MODERN DENGAN MENGGUNAKAN *FRAMEWORK FLUTTER* YANG MENTARGET *MULTI PLATFORM*

Proposal Skripsi

Disusun untuk memenuhi salah satu syarat memperoleh gelar Sarjana Komputer

Oleh:
Gian Chiesa Maghriza
1313618021

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

2022

LEMBAR PENGESAHAN

Dengan ini saya mahasiswa Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Jakarta

Nama	:	Lazuardy Khatulistiwa
No. Registrasi	:	1313618008
Jurusan	:	Ilmu Komputer
Judul	:	Perancangan Aplikasi *Search Engine* dengan Mengintegrasikan *Web Crawler*, Algoritma *Page Ranking*, dan *Document Ranking*

Menyatakan bahwa proposal skripsi ini telah siap diajukan untuk Seminar Pra Skripsi.

Menyetujui,

Dosen Pembimbing I	Dosen Pembimbing II
Muhammad Eka Suryana, M.Kom	**Drs. Mulyono, M.Kom**
NIP. 19851223 201212 1 002	NIP. 19660517 199403 1 003

Mengetahui,

Koordinator Program Studi Ilmu Komputer

Ir. Fariani Hermin M.T

NIP. 19600211 198703 2 001

LEMBAR PERNYATAAN

Saya menyatakan dengan sesungguhnya bahwa proposal skripsi dengan judul "**Perancangan Aplikasi *Search Engine* dengan Mengintegrasikan *Web Crawler*, Algoritma *Page Ranking*, dan *Document Ranking***" yang disusun sebagai syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Ilmu Komputer Universitas Negeri Jakarta adalah karya ilmiah saya dengan arahan dari dosen pembimbing.

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

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

Jakarta, 6 Juni 2022

Lazuardy Khatulistiwa

1313618008

HALAMAN PERSEMBAHAN

Untuk Keluargaku dan Diriku Sendiri.

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT, karena dengan rahmat dan karunia-Nya, penulis dapat menyelesaikan proposal skripsi yang berjudul **"Perancangan Arsitektur *Search Engine* dengan Mengintegrasikan *Web Crawler*, Algoritma *Page Ranking*, dan *Document Ranking*"**.

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

1. Yth. Para petinggi di lingkungan FMIPA Universitas Negeri Jakarta.
2. Yth. Ibu Ir. Fariani Hermin Indiyah, M.T selaku Koordinator Program Studi Ilmu Komputer.
3. Yth. Bapak Muhammad Eka Suryana, M.Kom selaku Dosen Pembimbing I yang telah membimbing, mengarahkan, serta memberikan saran dan koreksi terhadap proposal skripsi ini.
4. Yth. Bapak Drs. Mulyono, M.Kom selaku Dosen Pembimbing II yang telah membimbing, mengarahkan, serta memberikan saran dan koreksi terhadap proposal skripsi ini.
5. Ayah dan Ibu penulis yang selama ini telah mendukung dan membantu menyelesaikan proposal skripsi ini.
6. Teman-teman Program Studi Ilmu Komputer 2018 yang telah mendukung dan membantu proposal skripsi ini.

Penulis menyadari bahwa penyusunan proposal skripsi ini masih jauh dari sempurna karena keterbatasan ilmu dan pengalaman yang dimiliki. Oleh karenanya, kritik dan saran yang bersifat membangun akan penulis terima dengan senang hati. Akhir kata, penulis berharap tugas akhir ini bermanfaat bagi semua pihak khususnya penulis sendiri. Semoga Allah SWT senantiasa membalas kebaikan semua pihak yang telah membantu penulis dalam menyelesaikan proposal skripsi ini.

Jakarta, 6 Juni 2022

Lazuardy Khatulistiwa

DAFTAR ISI

LEMBAR PENGESAHAN **iii**

LEMBAR PERNYATAAN **iv**

HALAMAN PERSEMBAHAN **v**

KATA PENGANTAR **vii**

DAFTAR ISI **viii**

DAFTAR GAMBAR **x**

DAFTAR TABEL **xi**

I PENDAHULUAN **1**

A. Latar Belakang Masalah 1

B. Rumusan Masalah 6

C. Pembatasan Masalah 6

D. Tujuan Penelitian 6

E. Manfaat Penelitian 6

II KAJIAN PUSTAKA **8**

A. Konsep Budidaya Ikan Air Tawar 8

B. *Recirculating aquaculture system (RAS)* 8

C. *Biofloc Technology* 9

D. Dart 10

E. Flutter 12

F. Flutter Widget 13

G. Pengaplikasian REST API pada Flutter 22
H. Instalasi Flutter Untuk Pengembangan Aplikasi Android 25
I. *Preprocessing Data* 28
J. Pembobotan *TF-IDF* 30
K. *Google PageRank* 31
1. *Simplified PageRank* 34
2. *PageRank* 34
L. Metode Scrum 35
M. Tim Scrum 36
N. Aktivitas-aktivitas Scrum (*Scrum Events*) 39
O. Komponen Scrum 40

III METODOLOGI PENELITIAN **43**
A. Deskripsi Sistem 43
B. Desain Penelitian 44
C. Alat dan Bahan Penelitian 45
D. Perancangan Sistem 46
1. *Product Backlog* 46
2. *Sprint Backlog* 48
3. *Daily Scrum*, *Sprint Review*, dan *Sprint Retrospective* 49
4. *Sprint 1 Report* 49

LAMPIRAN **56**

A Transkrip Percakapan **56**

DAFTAR PUSTAKA **58**

DAFTAR GAMBAR

Gambar 1.1 Produksi *Tilapia* 1996-2017 (Sayed, 2020) 2

Gambar 1.2 Produksi *Tilapia* Di Pulau Jawa (KKP, 2021) 2

Gambar 1.3 Indeks Konsumsi Ikan Di Pulau Jawa (KKP, 2021) 3

Gambar 1.4 Perbandingan Tebar Padat Kolam (Setiawan, 2021) 4

Gambar 1.5 Form Pencatatan Kualitas Air 5

Gambar 2.1 Contoh Main Function 11

Gambar 2.2 Contoh Deklarasi Variable 11

Gambar 2.3 Contoh Deklarasi Variabel Secara Eksplisi 11

Gambar 2.4 Contoh Class Pada Dart 12

Gambar 2.5 Contoh Text Widget 14

Gambar 2.6 Contoh Textfield Widget 15

Gambar 2.7 Contoh Image Widget 16

Gambar 2.8 Contoh AppBar 16

Gambar 2.9 Item Item Pada AppBar 17

Gambar 2.10 Contoh Pengaplikasin AppBar 17

Gambar 2.11 Contoh Card 18

Gambar 2.12 Contoh Pengaplikasian ListView pada Card Widget 19

Gambar 2.13 Output Pengaplikasian ListView pada Card Widget 19

Gambar 2.14 Contoh tampilan bottom navigation bar 20

Gambar 2.15 Contoh code bottom navigation bar 21

Gambar 2.16 Contoh variabel 21

Gambar 2.17 Contoh Penerapan Fungsi setState 21

Gambar 2.18 Contoh cara merubah value currentIndex dengan variable . . 22

Gambar 2.19 Contoh Membuat *Network Request* 23

Gambar 2.20 Contoh Model Class 24
Gambar 2.21 Contoh Cara Konversi http.response 24
Gambar 2.22 Contoh Cara Memanggil *Factory Method*Ke *initState()* 25
Gambar 2.23 System Requirement instalasi Flutter 25
Gambar 2.24 Indikator Flutter Sudah Terinstal Pada Path 26
Gambar 2.25 Cara Menjalankan Flutter Doctor 26
Gambar 2.26 Contoh Output Flutter Doctor 27
Gambar 2.27 Tahapan *preprocessing data* 29
Gambar 2.28 Contoh *backlink* 33

Gambar 3.1 *Wireframe* Tampilan Sistem 43
Gambar 3.2 *Flowchart* Sistem 44
Gambar 3.3 Desain Penelitian 45
Gambar 3.4 *Github Projects Sprint-1* 50
Gambar 3.5 *Entity Relationship Diagram Sprint-1* 51
Gambar 3.6 *Class Diagram Sprint-1* 52
Gambar 3.7 Pengujian *crawler* penelitian sebelumnya 53
Gambar 3.8 Pengujian *crawler* versi terbaru 53
Gambar 3.9 Hasil pengujian *crawler* penelitian sebelumnya 54
Gambar 3.10 Hasil pengujian *crawler* versi terbaru 54
Gambar 3.11 Hasil pengujian *crawler* terbaru 2 55

DAFTAR TABEL

Tabel 3.1 *Product Backlog* 47

Tabel 3.2 *Sprint 1 Backlog* 48

BAB I

PENDAHULUAN

A. Latar Belakang Masalah

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

Diantara jenis ikan air tawar yang memiliki nilai ekonomi, Lele dan Nila adalah jenis ikan yang memiliki masa panen yang cenderung singkat. Masing-masing memiliki masa panen 3 bulan serta 5 bulan. Ikan Nila (*Tilapia*) memegang 8 persen dari total produksi ikan dunia dengan produksi meningkat dari tahun ke tahun (Sayed, 2020, hal tersebut dapat dilihat pada gambar 1.1. Sementara itu data KKP menunjukkan, volume produksi *Tilapia* di pulau jawa juga memiliki tren yang sama. Kecuali DKI Jakarta yang stagnant dan Jawa Barat yang menurun, seperti yang ditunjukan oleh gambar 1.2. DKI Jakarta tidak memiliki lahan pertanian yang tidak luas sehingga dapat dimaklumi, untuk Jawa Barat penyebab penurunnya masih perlu dihuhungkan dengan ekspor-impor *Tilapia*, dan jumlah pertumbuhan

rumah tangga perikanan (KKP, 2021). Ditinjau dari gambar 1.3, indeks konsumsi ikan di Jawa Barat trendnya mengalami peningkatan sehingga logika yang wajar dari turunnya jumlah produksi ikan di Jawa Barat adalah tidak terupdatenya data.

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

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

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

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

Gambar 1.3: Indeks Konsumsi Ikan Di Pulau Jawa (KKP, 2021)

Budidaya ikan air tawar di Indonesia secara umum terbagi menjadi 3 kelas, tetapi hanya dua yang dibahas di sini yaitu *Semi-Intensive* dan *Intensive*. Perbedaan dari keduanya terdapat pada kemampuan tebar padat ikan, dimana *Intensive aquaculture system* mampu mengakomodasi kepadatan ikan dengan luas penampang lahan yang sama. *Semi Intensive* salah satu contohnya pada sistem Tambak. Sementara *Intensive system* diwakili oleh *Biofloc Technology (BFT)* dan *Recirculating Aquaculture System (RAS)*. Ditinjau dari tingkat keahlian yang dibutuhkan *BFT* memerlukan tenaga ahli khusus.

BFT pertama kali diaplikasikan secara komersial oleh *Belize Aquaculture Ltd* pada tahun 1997. Pada uji coba yang dilakukan, produksi udang menggunakan *BFT* menghasilkan 11-26 Ton / Hektar. Jauh di atas tambak konvensional (Samocha, 2019). Hal ini disebabkan kualitas air yang dihasilkan pada *BFT* di atas kualitas air biasa terutama dalam 2 hal yakni, kekayaan dari bahan organik dalam air, dan kemamuan air dalam mereduksi *toxic* secara alami. *BFT* mendapatkan hal tersebut dengan bantuan *Microbacteria* dari keluarga *Bacillus sp* (Fattah, 2020). *RAS* memiliki kelebihan yaitu tidak dibutuhankan teknisi khusus kecuali saat instalasi kolam. Namun, *RAS* memiliki kemampuan tebar padat yang lebih rendah dibanding *BFT* sehingga skala produktivitasnya masih lebih rendah dibanding sistem *BFT*. Sistem *BFT* memiliki kemampuan tebar padat yang tinggi seperti yang di tunjukan

pada gambar 1.4. Namun demikian *BFT* memiliki kelemahan utama yaitu butuh tenaga ahli yang mumpuni atau rumah tangga yang mengaplikasikan *BFT* akan mengalami kegagalan.

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

Sudah banyak peneliti di Indonesia yang berkontribusi dalam berbagai teknologi *Aqua Culture*. (Supriati, 2018) meneliti dan mengembangkan Sistem Informasi Akuntansi Budidaya Perikanan berbasis *SAK EMKM* dan *Android*. Aplikasi yang dikembangkan diperuntukkan untuk Kelompok Tani yang tersusun atas Ketua, Sekretaris, Bendahara, dan Seksi Sarana Produksi perikanan. Aplikasi ini memfasilitasi sejumlah fitur seperti transaksi jual-beli, kas keluar-masuk, dan pembukuan. Penelitian yang sejenis dilakukan oleh (Widhiastika, 2021), aplikasi hasil penelitian memfasilitasi user untuk melakukan transaksi jual beli perikanan. Aplikasi yang dibuat dinamakan dengan FO-KLIK. Dikembangkan dengan *Android* yang dipasangkan dengan *Firebase*. Selain transaksi jual beli, FO-Klik turut menyediakan fasilitas diskusi dan informasi nilai gizi dari suatu produk item perikanan. Namun tidak dilengkapi dengan manajemen keuangan seperti yang

dilakukan oleh (Supriati, 2018). Selain dari penelitian diatas, perancangan aplikasi terkait budidaya ikan air tawar juga dapat dilakukan berdasarkan metode atau sistem budidaya yang digunakan. Oleh karena itu, penelitian pengembangan aplikasi terkait teknologi perikanan modern ini menarik untuk dilakukan.

Tenaga ahli bidang *BFT* melakukan pemeriksaan kualitas air yang ketat setiap hari. Pengukuran kualitas air ini dilakukan untuk setiap kolam budidaya yang ada seperti yang ditunjukkan pada Gambar 1.5. Hal ini tentunya berimplikasi pada pengelolaan arsip formuilir tersebut perlu dilakukan dengan benar dan rapi agar data yang dikumpulkan dapat dievaluasi sebagai dasar pertimbangan *treatment* yang akan dilakukan pada suatu kolam budidaya. Jika hal ini tidak dilakukan, maka berpotensi pada kegagalan proses budidaya berbasis *BFT*. Potensi kegagalan tersebut juga diperbesar dengan metode pencatatan yang dilakukan masih berbasis kertas.

AKTIVITAS KOLAM BUDIDAYA
BIOFLOC IKAN NILA

Pemeliharaan : Jenis ikan : Jumlah ekor : Kolam :

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

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

Gambar 1.5: Form Pencatatan Kualitas Air

Berdasarkan latar belakang yang telah dijelaskan, Peneliti mengusulkan pengembangan aplikasi manajemen budidaya perikanan *BFT* yang dimana sistem

tersebut memiliki skala produktivitas yang lebih tinggi dibanding dengan sistem *RAS*. Aplikasi ini berpotensi meningkatkan jumlah ikan yang dipanen dan menurunkan angka kematian sehingga mampu menurunkan harga jual ikan dengan adanya stok melimpah. Dengan demikian, dapat terwujud ketahanan pangan wilayah. Aplikasi ini diharapkan dapat membantu petani yang menerapkan konsep budidaya ikan modern berbasis *BFT*.

B. Rumusan Masalah

Dari uraian latar belakang di atas, perumusan masalah pada penelitian ini ialah "Bagaimana perancangan aplikasi pendukung teknologi perikanan modern dengan menggunakan *framework Flutter* yang mentarget *multi platform*?"

C. Pembatasan Masalah

Pembatasan masalah pada penelitian ini antara lain:

1. Aplikasi dikembangkan hanya untuk 1 user saja.

D. Tujuan Penelitian

1. Membuat aplikasi yang memudahkan pengelolaaan manajemen data *aqua culture BFT*.
2. Meningkatkan hasil panen dan menekan angka kematian ikan saat proses budidaya dengan aplikasi tersebut.

E. Manfaat Penelitian

1. Bagi penulis

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

2. Bagi Program Studi Ilmu Komputer

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

3. Bagi Universitas Negeri Jakarta

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

BAB II

KAJIAN PUSTAKA

A. Konsep Budidaya Ikan Air Tawar

Budidaya adalah kegiatan memproduksi dan mengembangkan biota (organisme) dalam lingkungan yang terkendali untuk mendapatkan keuntungan. Budidaya ikan air tawar merupakan kegiatan yang dilakukan untuk meningkatkan produktivitas perairan, khususnya ikan air tawar. Kegiatan budidaya dimaksudkan untuk memperbanyak, menumbuhkan dan meningkatkan kualitas biota perairan itu sendiri untuk menghasilkan keuntungan (Bangkit, 2016).

Akuakultur (budidaya ikan) merupakan salah satu subsektor yang diharapkan dapat mewujudkan misi mensejahterakan perikanan. Akuakultur tingkat rendah berkontribusi pada kesejahteraan petani ikan untuk menjamin ketersediaan pangan, gizi dan kesehatan di daerah (Edwards dan Demain, 1998).

B. *Recirculating aquaculture system (RAS)*

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

negara maju, seperti Amerika, Israel, Singapura, German serta Norwegia selama kurun waktu 20-30 tahun ini (Fadhil, 2010).

Teknologi RAS menawarkan sebuah alternatif teknologi budidaya melalui perbaikan kualitas air dan penggunaan kembali (re-use). Penggunaan RAS secara intensif terbukti dapat mengurangi secara signifikan konsumsi air dan konsentrasi nutrien melalui perbaikan dan pengembangan teknologi secara berkelanjutan (Thesiana dan Pamungkas, 2015). RAS dapat digunakan untuk mengontrol beberapa parameter kualitas air penting seperti oksigen terlarut, karbondioksida, amonia, nitrit, nitrat, pH, salinitas, dan padatan tersuspensi. Hal ini memungkinkan terciptanya kondisi pemeliharaan yang baik untuk pertumbuhan dan pemanfaatan pakan yang lebih optimal (Dalsgaard et al., 2013)

C. *Biofloc Technology*

Teknologi bioflok merupakan teknologi penggunaan bakteri baik heterotrof maupun autotrof yang dapat mengonversi limbah organik secara intensif menjadi kumpulan mikroorganisme yang berbentuk flok, kemudian dapat dimanfaatkan oleh ikan sebagai sumber makanan (de Schryver, 2009) (Avnimelech, 2012). Di dalam flok terdapat beberapa organisme pembentuk seperti bakteri, plankton, jamur, alga, dan partikel tersuspensi yang memengaruhi struktur dan kandungan nutrisi bioflok, namun komunitas bakteri merupakan mikroorganisme paling dominan dalam pembentukan flok dalam bioflok (de Schryver et al. 2008). Beberapa jenis bakteri yang sering digunakan dalam bioflok adalah Bacillus sp., Bacillus subtilis, Pseudomonas sp., Bacillus lichenoformis, Bacillus pumilus (Zao et al, 2012), Lactobacillus sp. (Anand et al, 2014), Bacillus megaterium (Suprapto, 2013).

Pada sistem akuakultur dengan teknologi bioflok, air media kultur hanya sekali dimasukkan dalam wadah, dan digunakkan sampai panen. Penambahan air

hanya untuk pengganti penguapan dan pengontrolan kepadatan bioflok (Avnimelech, 2009). Dibanding sistem resirkulasi yang sangat kompleks, sistem kultur dengan teknologi bioflok hanya menggunakan satu wadah, yakni wadah kultur. Penguraian bahan organik oleh bakteri dan mikroorganisme pengurai, sampai pada pemanfaatan hasil penguraian oleh mikroalga dan mikroorganisme yang tumbuh, terjadi dalam wadah secara seimbang dengan kepadatan organisme kultur yang sangat tinggi. Pengontrolan kualitas air terjadi dalam wadah kultur itu sendiri, oleh sistem bioflok yang sudah berjalan dalam wadah kultur. Sistem ini sangat murah, sederhana ramah lingkungan dan memiliki produktifitas yang sangat tinggi (Taw, 2014).

D. Dart

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

1. *Main Function*

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

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

Gambar 2.1: Contoh Main Function

2. Deklarasi *Variables*

   Berikut adalah contoh deklarasi Variable pada dart:

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

Gambar 2.2: Contoh Deklarasi Variable

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

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

Gambar 2.3: Contoh Deklarasi Variabel Secara Eksplisi

3. *Classes*

   Dart adalah bahasa berorientasi objek dengan Class dan pewarisan berbasis mixin. Setiap objek adalah turunan dari Class, dan semua Class kecuali Null turun dari Objek. Pewarisan berbasis mixin berarti meskipun setiap Class (kecuali Top-Class) memiliki tepat satu superclass, Class dapat digunakan kembali dalam beberapa hierarki Class. Metode Extendsi adalah cara untuk

menambahkan fungsionalitas ke Class tanpa mengubah Class atau membuat subclass. Dapat Dilihat pada gambar 2.4 yaitu contoh sebuah Class pada bahasa pemrogramman Dart.

```
class Spacecraft {
  String name;
  DateTime? launchDate;

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

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

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

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

Gambar 2.4: Contoh Class Pada Dart

E. Flutter

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

mengubah tampilan UI sesuai dengan yang diinginkan oleh developer.

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

F. Flutter Widget

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

1. Widget digunakan untuk membuat UI, seperti: baris, kolom, stack, card, form, dan padding.
2. Pada Widget, developer dapat melakukan styling, seperti: tipe font, ukuran, berat, warna, batas, dan bayangan.
3. Widget dapat berupa kumpulan bentuk dan formulir.

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

1. *Text Widget*

Teks adalah bagian yang sangat penting dari sebuah UI aplikasi mobile mana pun. Pada Flutter, untuk menampilkan text tersebut, dapat menggunakan Text widget seperti pada gambar 2.5.

Gambar 2.5: Contoh Text Widget

2. *Textfield Widget*

Textfield adalah sebuah widget yang dapat digunakan pengguna untuk mengetik sebuah text kedalam aplikasi. Biasanya Textfield digunakan untuk berbagai fungsi aplikasi seperti sebagai sebuah bagian dari form, sebagai alat searching dalam aplikasi, mengetik pesan kedalam aplikasi dan masih banyak lagi. Pada gambar 2.6 diperlihatkan bagaimana cara untuk membuat sebuah textfield.

Gambar 2.6: Contoh Textfield Widget

3. *Image Widget*

Image Widget adalah sebuah widget yang berfungsi untuk menampilkan gambar pada aplikasi. Flutter dapat mensupport image dengan berbagai macam format atau ekstensi seperti JPG, PNG, GiF, WebP dan masih banyak lagi. image widget juga dapat diaplikasikan kedalam widget lain seperti card, dan lainnya sebagai childern. Berikut adalah contoh penggunaan image wdget pada flutter yang dapat dilihat pada gambar 2.7.

Gambar 2.7: Contoh Image Widget

4. *AppBar*

AppBar merupakan salah satu bagian yang umumnya digunakan pada sebuah aplikasi sebagai menu petunjuk untuk memudahkan pengguna aplikasi. Widget ini terletak pada posisi bagian paling atas aplikasi yang tampilannya kurang lebih seperti pada gambar 2.8.

Gambar 2.8: Contoh AppBar

Sebuah AppBar dapat berisi beberapa toolbar yang biasanya merupakan sebuah widget juga yang digunakan untuk membantu user mengakses berbagai fungsi dalam aplikasi. diperlihatkan pada gambar 2.9 mengenai item-item atau tool yang terdapat pada AppBar. Dan untuk cara pengaplikasian AppBar dapat dilihat pada gambar 2.10.

Gambar 2.9: Item Item Pada AppBar

Gambar 2.10: Contoh Pengaplikasin AppBar

5. *Card Widget*

Card adalah sebuah panel yang biasa diguanakan pada aplikasi untuk merepresentasikan informasi yang berkaitan dengan konten pada aplikasi. Informasi yang ditampilkan dapat berupa judul, gambar, tombol, dan lainnya.

```
@override
Widget build(BuildContext context) {
  return Center(
    child: Card(
      child: InkWell(
        splashColor: Colors.blue.withAlpha(30),
        onTap: () {
          debugPrint('Card tapped.');
        },
        child: const SizedBox(
          width: 300,
          height: 100,
          child: Text('A card that can be tapped'),
        ),
      ),
    ),
  );
}
}
```

Gambar 2.11: Contoh Card

6. List View

List View adalah bentuk default dari sebuah ListView class. Dengan ListView maka children atau widget yang ada di dalamnya akan menjadi scrollable (bisa di scroll). Penggunaan default ListView ini hanya untuk widget yang bersifat statis. Statis yang dimaksud bukan untuk isi kontennya melainkan lebih kepada jumlah widget di dalamnya.Untuk list yang bersifat dinamis (jumlah list item mengikuti dari jumlah data) maka gunakan ListView.builder. ListView.builder memiliki dua properti utama yaitu itemCount (jumlah list item) dan itemBuilder (untuk membangun tampilan dari list item). Penggunaan List View juga dapat diterapkan pada Card Widget sehingga card tersebut dapat memuat sejumlah informasi mengikuti jumlah data yang kita punya. Secara sederhana penggunaan listview.builder sebagai berikut:

```
body: ListView.builder(
  itemBuilder: (context, index) {
    return Card(
      child: ListTile(
          title: Text(
            bulan[index],
            style: TextStyle(fontSize: 30)
          ),
          subtitle: Text('ini subtitle dari ' + bulan[index]),
          leading: CircleAvatar(
            child: Text(
              bulan[index][0], // ambil karakter pertama text
              style: TextStyle(fontSize: 20)
            ),
          )
      ),
    );
  },
  itemCount: bulan.length,
),
```

Gambar 2.12: Contoh Pengaplikasian ListView pada Card Widget

Dapat dilihat pada gambar menggunakan data berisikan nama bulan dari januari sampai desember sebagai data yang akan di gunakan untuk List View yang akan menghasilkan output tampilan seperti gambar 2.3.

Gambar 2.13: Output Pengaplikasian ListView pada Card Widget

7. Bottom Navigation Bar

Bottom Navigation Bar adalah sebuah widget yang terdiri atas beberapa text label, icon, atau keduanya yang berada pada bagian bawah aplikasi. Bottom Navigation Bar menyediakan sebuah akses cepat antara top-level screen view dari sebuah aplikasi. Untuk membuat Bottom Navigation Bar, pada aplikasi flutter umumnya menggunakan bottomNavigationBar widget. Widget ini merupakan widget yang bersifat dinamis jadi baiknya menggunakan statefull widget. Dan karena untuk satu bottomNavigationBar dapat memiliki lebih dari satu widget maka widget ini juga dapat dikategorikan sebagai multiple children widget.

Gambar 2.14: Contoh tampilan bottom navigation bar

Pada BottomNavigationBar terdapat properti items untuk menempatkan item widget, currentIndex untuk menentukan item mana yang sedang aktif lalu properti lainnya untuk warna dan memunculkan label yang dapat dilihat pada gambar 2.5. Selanjutnya agar selected item menjadi dinamis maka perlu merubah nilai 0 pada currentIndex menjadi variable dengan tipe integer dengan nilai 0 sebagai default value seperti gambar 2.6.

```
bottomNavigationBar: BottomNavigationBar(
  items: const <BottomNavigationBarItem>[
    BottomNavigationBarItem(
      icon: Icon(Icons.home),
      title: Text('Beranda'),
    ),
    BottomNavigationBarItem(
      icon: Icon(Icons.assignment),
      title: Text('Pesanan'),
    ),
    BottomNavigationBarItem(
      icon: Icon(Icons.mail),
      title: Text('Inbox'),
    ),
    BottomNavigationBarItem(
      icon: Icon(Icons.person),
      title: Text('Akun'),
    ),
  ],
  currentIndex: 0,
  selectedItemColor: Colors.green,
  unselectedItemColor: Colors.grey,
  showUnselectedLabels: true,
),
```

Gambar 2.15: Contoh code bottom navigation bar

```
int _selectedNavbar = 0;
```

Gambar 2.16: Contoh variabel

Lalu buat fungsi untuk memanggil fungsi setState yang berguna untuk mengupdate nilai selectedNavbar seperti gambar 2.7.

```
void _changeSelectedNavBar(int index) {
  setState(() {
    _selectedNavbar = index;
  });
}
```

Gambar 2.17: Contoh Penerapan Fungsi setState

Setelah itu tambah property onTap dan isi valuenya dengan nama fungsi yang tadi dibuat. Ubah value currentIndex menjadi variable seperti gambar 2.8.

```
...
showUnselectedLabels: true,
currentIndex: _selectedNavbar,
onTap: _changeSelectedNavBar,
```

Gambar 2.18: Contoh cara merubah value currentIndex dengan variable

Dalam framework Flutter terdapat fitur yang bernama hot-reload. Dengan fitur ini, developer membangun User Interface (UI), menambahkan fitur, dan memperbaiki bug dapat menjadi lebih cepat dan mudah. Hot-reload bekerja dengan cara menanamkan kode file baru kedalam Dart Virtual Machine (VM). Setelah VM diperbaharui, kerangka kerja pada Flutter membuat ulang semua pohon widget secara otomatis. Hal ini membuat developer untuk melihat perubahan langsung pada UI aplikasi yang dikembangkannya.

Terdapat dua kombinasi class dalam mengembangkan aplikasi menggunakan framework Flutter, yaitu: Stateless Widgets dan Statefull Widgets. Class tersebut merupakan widget yang akan menggambarkan tampilan User Interface dengan membangun konstelasi widget lainnya agar menjadi lebih konkret. Stateless Widget hanya dapat menggambar satu kali saat widget dimuat sehingga tidak dapat menggunakan fitur hot-reload karena state tersebut sifatnya statis. Sebaliknya, Statefull Widgets dapat menggunakan hot-reload karena sifatnya yang dinamis. Dalam Statefull Widgets terdapat siklus hidup yang penting untuk diketahui oleh developer Flutter.

G. Pengaplikasian REST API pada Flutter

Representational state transfer (REST) diciptakan oleh Roy Fielding dari University of California. Ini adalah layanan web yang sangat sederhana dan ringan dibandingkan dengan SOAP. Kinerja, skalabilitas, kesederhanaan, portabilitas, dan kemampuan modifikasi adalah prinsip utama di balik desain REST. REST API

memungkinkan berbagai sistem untuk berkomunikasi dan mengirim/menerima data dengan cara yang sangat sederhana. Setiap panggilan API, REST memiliki hubungan antara kata kerjaHTTP dan URL. Sumber daya dalam database dalam suatu aplikasi dapat dipetakan dengan titik akhir API di REST.

REST bekerja dengan bernavigasi melalui link-link HTTP dalam aktivitas tertentu, sehingga seperti terjadi perpindahan antara state. Beberapa perintah HTTP antara lain fungsi GET, POST, PUT, UPDATE atau DELETE. Supaya informasi yang diterima lebih mudah dibaca dan di-parsing disisi client, maka balasan yang dikirimkan oleh API server dalam bentuk JSON.

Dalam pengaplikasian REST API, Dart dan Flutter menyediakan alat seperti http package, untuk jenis pekerjaan ini. Berikut adalah langkah langkah untuk menerapkannya:

1. Tambahkan *http package*

   Untuk instalasi *http package*, tambahkan *http package* pada dependecies section di file pubspec.yml. Lalu tambahkan internet permission pada file AndroidManifest.xml. Terakhir, import http package pada class yang akan diaplikasikan

2. Membuat *Network Request*

```
Future<http.Response> fetchAlbum() {
  return http.get(Uri.parse('https://jsonplaceholder.typicode.com/albums/1'));
}
```

Gambar 2.19: Contoh Membuat *Network Request*

Berikut adalah contoh bagaimana membuat request dan mengambil data dar JSONPlacehoder menggunakan http.get() method yang dimana http.get() mengembalikan *Future* yang memuat response. *Future* merupakan class dari

Dart yang berkerja secara asynchronus. Lalu http.Response memuat data yang diterima dari pemanggilan http yang sukses.

3. Mengubah *Response* menjadi *Custom Dart Object*

   Buat Album class yang memuat data dari request termasuk juga membuat constructornya. berikut adalah contohnya

```
class Album {
  final int userId;
  final int id;
  final String title;

  const Album({
    required this.userId,
    required this.id,
    required this.title,
  });

  factory Album.fromJson(Map<String, dynamic> json) {
    return Album(
      userId: json['userId'],
      id: json['id'],
      title: json['title'],
    );
  }
```

Gambar 2.20: Contoh Model Class

Setelah itu konversi http.response menjadi sebuah Album. Konversi response body menjadi Json Map menggnakan Dart Convert package. jika server mengembalikan OK ressponse dengan kode status 200, konversi Json Map menjadi Album menggunakan method fromJson().

```
Future<Album> fetchAlbum() async {
  final response = await http
      .get(Uri.parse('https://jsonplaceholder.typicode.com/albums/1'));

  if (response.statusCode == 200) {
    // If the server did return a 200 OK response,
    // then parse the JSON.
    return Album.fromJson(jsonDecode(response.body));
  } else {
    // If the server did not return a 200 OK response,
    // then throw an exception.
    throw Exception('Failed to load album');
  }
}
```

Gambar 2.21: Contoh Cara Konversi http.response

4. Ambil dan Tampilkan Data

   Ambil data dengan cara memanggil method yang sudah dibuat kedalam initState()

```
class _MyAppState extends State<MyApp> {
  late Future<Album> futureAlbum;

  @override
  void initState() {
    super.initState();
    futureAlbum = fetchAlbum();
  }
  // ...
}
```

Gambar 2.22: Contoh Cara Memanggil *Factory Method*Ke *initState()*

Terakhir, tampilan data tersebut kedalam UI widget.

H. Instalasi Flutter Untuk Pengembangan Aplikasi Android

Flutter adalah cross-platform framework yang dibuat oleh Google untuk membangun berbagai aplikasi seperti android mobile, iOS mobile, web, dan desktop. Sebelum memulai proses pengembangan sebuah aplikasi android dengan flutte, maka perlu diketahui terlebiuh dahulu terkait langkah dalam instalasi flutter. Berikut adalah langkah-langkah instalasi flutter untuk pengembangan aplikasi android:

1. *System Requirements*

   Untuk instalasi dan mengembangkan flutter, dibutuhkan beberapa requirements yang harus dipenuhi dalam lingkungan pengembangan seperti yang di tunjukan pada gambar 2..

- **Operating Systems**: Windows 10 or later (64-bit), x86-64 based.
- **Disk Space**: 1.64 GB (does not include disk space for IDE/tools).
- **Tools**: Flutter depends on these tools being available in your environment.
  - Windows PowerShell 5.0 or newer (this is pre-installed with Windows 10)
  - Git for Windows 2.x, with the **Use Git from the Windows Command Prompt** option.

Gambar 2.23: System Requirement instalasi Flutter

2. Unduh Flutter SDK

   Flutter Software Development Kit (SDK) dapat diunduh melalui website resmi flutter. Pastikan unduh dokumen tersebut dengan versi terbaru. Setelah SDK diunduh, ekstrak file tersebut. Ada beberapa syarat dalam menempatkan hasil ekstrak flutter SDK yaitu: Flutter SDK tidak boleh ditempatkan pada folder yang namanya mengandung spasi, lalu tidak disarankan untuk menginstal Flutter kedalam Folder yang memerlukan privilege tinggi seperti dalam folder Program Files pada Local Disk C.

3. Update Path

   Untuk dapat Menjalankan perintah Flutter seperti biasa pada Windows Console, maka diperlukan langkah untuk menambahkan Flutter kedalam Path Environment Variable. Tambahkan Directory Flutter/bin kedalam variable Path sebagai value. Setelah menambahkan Flutter kedalam Path Environment Variable, pastikan kembali jika path sudah terupdate dengan benar menggunakan perintah 'where flutter dart' pada Windows Command Prompt. Jika sudah berhasil, akan dihasilkan output seperti gambar 2.x pada Windows Console.

```
C:\>where flutter dart
C:\dev\src\flutter\bin\flutter
C:\dev\src\flutter\bin\flutter.bat
C:\dev\src\flutter\bin\dart
C:\dev\src\flutter\bin\dart.bat
C:\dev\src\dart-sdk\bin\dart.exe
```

Gambar 2.24: Indikator Flutter Sudah Terinstal Pada Path

4. Jalankan Flutter Doctor

```
C:\src\flutter>flutter doctor
```

Gambar 2.25: Cara Menjalankan Flutter Doctor

Jalankan perintah Flutter Doctor untuk melihat apakah ada dependensi platform yang masih diperlukan untuk menyelesaikan instalasi. Perintah ini memeriksa Environment komputer dan menampilkan laporan status penginstalan Flutter pada komputer. Periksa output dengan hati-hati untuk memastikan apakah masih ada perangkat lunak lain yang mungkin perlu diinstal atau task lebih lanjut yang harus dilakukan seperti pada gambar 2.x (ditampilkan dalam teks tebal).

```
[-] Android toolchain - develop for Android devices
    • Android SDK at D:\Android\sdk
    ✗ Android SDK is missing command line tools; download from https://goo.gl/XxQghQ
    • Try re-installing or updating your Android SDK,
      visit https://docs.flutter.dev/setup/#android-setup for detailed instructions.
```

Gambar 2.26: Contoh Output Flutter Doctor

5. Instalasi Android SDK

   Android SDK bisa didapatkan dengan menginstal Android Studio. Setelah menginstal Android Studio, pergi ke bagian Android studio Setup Wizard. Terakhir Instal plugin-plugin yang dibutuhkan untuk mengembangkan aplikasi Android pada Flutter seperti Android SDK, Android SDK command-line tools, dan Android SDK build tool.

6. Set up Android Emulator

   Android Emulator diperlukan untuk mencoba dan menjalankan aplikasi kita saat proses pengembangan. Android Studio dapat digunakan untuk membuat emulator atau virtual device yang dibutuhkan untuk menjalankan aplikasi. Buka Android Studio dan pergi menuju menu AVD Manager untuk membuat virtual device.

I. *Preprocessing Data*

Setiap dokumen atau data seringkali memiliki struktur yang berubah-ubah atau tidak terstruktur. Oleh karena itu, diperlukan suatu proses yang dapat mengubah bentuk data yang sebelumnya tidak terstruktur menjadi data terstruktur. Proses konversi ini disebut *text processing* (Feldman dan Sanger, 2007).

Data teks tersebut berisi banyak variasi dari mulai huruf hingga tanda baca. Variasi huruf harus konsisten, yaitu hanya menggunakan huruf besar saja atau huruf kecil saja. Selain itu, tanda baca juga perlu dihilangkan untuk mengurangi *noise* selama pencarian informasi.

Preprocessing data dilakukan agar data yang digunakan bebas noise, memiliki ukuran yang lebih kecil, dan lebih terorganisir sehingga dapat diproses lebih lanjut. Ada beberapa proses dalam tahap *preprocessing data* yaitu *case folding*, *tokenizing*, *stop word removal* dan *stemming* seperti pada gambar 2.27.

Gambar 2.27: Tahapan *preprocessing data*

Proses pertama pada tahap *preprocessing* adalah *case folding*. Tujuan dari proses case folding adalah untuk menyamakan karakter pada data. Proses ini mengubah seluruh huruf pada data menjadi huruf kecil. Karakter-karakter huruf besar dari A sampai Z yang terdapat pada data diubah kedalam karakter huruf kecil a sampai z. Sedangkan karakter-karakter selain huruf seperti tanda baca dan angka akan dihilangkan dari data dan dianggap sebagai *delimeter* atau batas pemisah (Manning dkk., 2009).

Sebelum data teks dapat diproses lebih lanjut, data yang tadinya terdiri dari banyak kalimat harus dipecah menjadi kata-kata terpisah, yang mana proses ini disebut tokenization. Strategi umum yang dilakukan selama proses tokenization adalah memisahkan kata dengan *whitespace* sebagai acuan pembatasnya dan menghilangkan tanda baca. Potongan-potongan kata yang sudah dihasilkan disebut

dengan *token*.

Tahap selanjutnya setelah *tokenizing* adalah *stop word removal* dan *stemming*. *Stop word removal* adalah pengambilan kata-kata penting dengan dari *token* dengan cara membuang kata yang kurang penting atau tidak bermakna. Contoh *stop word* dalam bahasa Indonesia dan akan dibuang dalam proses ini adalah "yang", "dan", "di", dll. Sedangkan *stemming* adalah proses penghilangan imbuhan pada sebuah kata sehingga menjadi kata dasar.

J. Pembobotan *TF-IDF*

Metode *Term Frequency-Inverse Document Frequency TF-IDF* adalah salah satu metode yang paling umum digunakan untuk menghitung bobot setiap kata dalam pencarian informasi. Cara ini juga dinilai efisien, sederhana dan memiliki hasil yang akurat.

TF-IDF adalah metode pemberian bobot pada hubungan antara kata (*word*) dan dokumen. *TF-IDF* adalah ukuran statistik yang digunakan untuk mengevaluasi pentingnya sebuah kata dalam dokumen atau kumpulan kata. Frekuensi kata dalam dokumen tertentu menunjukkan pentingnya dalam dokumen. Frekuensi dokumen yang berisi kata tersebut menunjukkan seberapa umum kata tersebut. Jika sering muncul dalam satu dokumen, bobot kata lebih besar, dan jika muncul di banyak dokumen, bobot kata akan lebih kecil.

Term Frequency (TF) menghitung berapa kali sebuah kata muncul dalam dokumen. Karena panjang setiap dokumen bisa berbeda, nilai *TF* dibagi dengan panjang dokumen (jumlah kata dalam dokumen). Berikut rumus dari *Term Frequency (TF)*:

$$tf_{t,d} = \frac{Number\ of\ times\ term\ t\ appears\ in\ a\ document}{Total\ number\ of\ terms\ in\ document} \tag{2.1}$$

Di mana maksud dari pembilang pada persamaan di atas adalah jumlah kata t muncul dalam satu dokumen dan maksud dari penyebutnya adalah jumlah semua kata yang ada dalam satu dokumen tersebut.

Setelah menghitung *Term Frequency (TF)*, selanjutnya adalah menghitung nilai *Inverse Document Frequency (IDF)*, yang merupakan ukuran seberapa penting suatu kata. *IDF* akan memberikan nilai pada kata-kata yang biasanya muncul sebagai kata yang kurang penting berdasarkan kemunculannya di seluruh dokumen. Semakin kecil nilai IDF, semakin tidak penting kata tersebut, dan sebaliknya. Berikut rumus dari *Inverse Document Frequency (IDF)*:

$$idf_t = log(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ term\ t\ in\ it}) \tag{2.2}$$

Di mana maksud dari pembilang pada persamaan di atas adalah jumlah dokumen yang ada pada dataset dan maksud dari penyebutnya adalah jumlah dokumen pada dataset yang terdapat kata t di dalamnya.

Nilai idf dari kata yang kemunculannya jarang akan bernilai tinggi, sedangkan nilai idf dari kata yang kemunculannya sering nilainya akan cenderung rendah (Manning dkk., 2009).

Setelah mempunyai nilai *TF* dan *IDF*, perhitungan bobot *TF IDF* adalah hasil perkalian dari nilai *TF* dan *IDF*.

$$tfidf_{t,d} = tf_{t,d} \times idf_t \tag{2.3}$$

K. *Google PageRank*

Setiap kali melakukan pencarian di *Google*, terdapat hasil *website* yang muncul di halaman secara berurutan, sebagian itu adalah karena *Google PageRank*.

Sederhananya, *PageRank* menganalisis jumlah tautan masuk ke setiap halaman di web, dan menetapkan setiap halaman sebuah peringkat yang ditentukan oleh peringkat setiap halaman yang menautkannya. *PageRank* mengikuti model probabilistik dari seorang peselancar web, yang mengunjungi halaman tertentu, dan kemudian mengklik link di halaman itu secara acak. Intinya, peringkat setiap halaman mewakili kemungkinan bahwa setiap peselancar web akan mendarat di sana. Meskipun secara konseptual sederhana, *PageRank* secara luas dianggap sebagai salah satu algoritma terpenting yang pernah ditemukan di industri teknologi.

Meskipun *PageRank* secara populer dikaitkan dengan Larry Page dan Sergey Brin, pendiri *Google*, versi awal masalah *eigenvalue* dieksplorasi pada tahun 1976 oleh Gabriel Pinski dan Francis Narin dalam upaya mereka untuk menentukan peringkat jurnal ilmiah. Thomas Saaty mengeksplorasi masalah yang sama pada tahun 1977 dalam konsepnya tentang *Analytic Heirarchy Process* yang menimbang keputusan, dan Bradley Love dan Steven Sloman menggunakannya pada tahun 1995 sebagai model kognitif untuk konsepnya juga.

Namun, baru pada tahun 1996 Larry Page dan Sergey Brin mengadopsi algoritma ini untuk digunakan dalam mesin pencari. Sebagai bagian dari proyek penelitian Stanford tentang penerapan mesin pencari jenis baru, Sergey Brin mengkonseptualisasikan gagasan untuk memeringkat halaman web dengan jumlah tautan yang mengarah ke sana. Makalah pertama mereka tentang masalah ini diterbitkan pada tahun 1998, yang mencakup prototipe awal mesin pencari Google, dan Page dan Brin juga mematenkan prosesnya (Universitas Stanford memegang paten, tetapi memberikan hak lisensi eksklusif kepada Google dengan imbalan 1,8 juta saham darinya). Sejak saat itu, algoritma ini tetap menjadi komponen integral dari mesin telusur *Google* selama bertahun-tahun, meskipun ini bukan satu-satunya kriteria yang digunakan *Google* untuk menentukan peringkat hasil penelusurannya.

Sebagai algoritma yang berlaku untuk *graph* apa pun secara umum, *PageRank* telah digunakan di banyak aplikasi lain selain mesin pencari, termasuk analisis jaringan jalan, ilmu saraf, bibliometrik, sastra, dan bahkan olahraga. Dalam industri teknologi, varian dari algoritma *PageRank* tetap digunakan sampai sekarang; *Twitter* menggunakannya untuk menyarankan pengguna untuk mengikuti, dan *Facebook* mengimplementasikannya di belakang layar sebagai bagian dari algoritma teman yang disarankan.

Algoritma *PageRank*, salah satu yang paling banyak digunakan terkait algoritma peringkat halaman, menyatakan bahwa jika suatu halaman memiliki *link* penting ke sana, link ke halaman lain juga menjadi penting. Oleh karena itu, *PageRank* memperhitungkan *backlink* dan menyebarkan peringkat melalui tautan, yaitu sebuah halaman memiliki peringkat tinggi jika jumlah peringkat *backlink*-nya tinggi. Gambar 2.28 menunjukkan contoh dari *backlink*: halaman A adalah *backlink* halaman B dan halaman C sedangkan halaman B dan halaman C adalah *backlink* dari halaman D.

Gambar 2.28: Contoh *backlink*

1. *Simplified PageRank*

Versi *PageRank* yang sedikit disederhanakan didefinisikan sebagai (Page dkk., 1999):

$$PR(u) = c \sum_{v \in B(u)} \frac{PR(v)}{N_v} \tag{2.4}$$

Di mana u merepresentasikan halaman web. $B(u)$ adalah kumpulan halaman yang merujuk ke u. $PR(u)$ dan $PR(v)$ masing-masing merupakan skor peringkat halaman u dan v. N_v menunjukkan jumlah tautan keluar dari halaman v. Sedangkan c adalah konstanta yang digunakan untuk normalisasi dan biasanya bernilai 0.85 namun dapat diubah sesuai keinginan.

Di *PageRank*, skor peringkat halaman, p, dibagi rata di antara tautan keluarnya. Nilai yang ditetapkan ke tautan keluar halaman p selanjutnya digunakan untuk menghitung peringkat halaman yang ditunjuk oleh halaman p. Skor peringkat halaman situs web dapat dihitung secara iteratif mulai dari halaman web mana pun. Dalam sebuah situs web, dua atau lebih halaman mungkin terhubung satu sama lain untuk membentuk satu lingkaran. Jika halaman ini tidak merujuk tetapi dirujuk oleh halaman web lain di luar *loop*, mereka akan mengakumulasi peringkat tetapi tidak pernah mendistribusikan peringkat apa pun. Skenario ini disebut *rank sink* (Page dkk., 1999).

2. *PageRank*

Untuk mengatasi masalah *rank sink*, diamati aktivitas pengguna. Sebuah fenomena ditemukan bahwa tidak semua pengguna mengikuti *link* yang ada. Misalnya, setelah melihat halaman a, beberapa pengguna mungkin tidak memutuskan untuk mengikuti tautan yang ada tetapi langsung menuju ke halaman b,

yang tidak langsung terhubung ke halaman a. Untuk tujuan ini, pengguna cukup mengetikkan *URL* halaman b ke dalam kolom teks URL dan langsung melompat ke halaman b. Dalam hal ini, peringkat halaman b harus dipengaruhi oleh halaman a meskipun kedua halaman ini tidak terhubung langsung. Oleh karena itu, tidak ada *rank sink* yang absolut. Berdasarkan pertimbangan fenomena tersebut di atas, PageRank yang original diterbitkan (Page dkk., 1999):

$$PR(u) = (1 - d) + d \sum_{v \in B(u)} \frac{PR(v)}{N_v} \tag{2.5}$$

Di mana d merupakan *damping factor* yang biasanya bernilai 0.85. Kita juga bisa menganggap d sebagai probabilitas pengguna mengikuti tautan dan dapat menganggap $(1 - d)$ sebagai distribusi *pagerank* dari halaman yang tidak terhubung secara langsung.

Untuk menguji kegunaan algoritma *PageRank*, *Google* menerapkannya pada mesin pencari *Google*. Dalam percobaan, algoritma *PageRank* bekerja secara efisien dan efektif karena nilai peringkat konvergen ke toleransi yang wajar dalam logaritma kasar $(log\, n)$ (Page dkk., 1999). Skor peringkat halaman web dibagi secara merata di atas halaman yang ditautkannya. Meskipun algoritma *PageRank* berhasil digunakan di *Google*, terdapat satu masalah yang masih ada: di web yang sebenarnya, beberapa tautan di halaman web mungkin lebih penting daripada yang lain (Xing dan Ghorbani, 2004).

L. Metode Scrum

Metode Scrum diciptakan oleh Jeff Sutherland dan tim pengembangnya pada awal 1990-an. Sutherland, Viktorov, Blount dan Puntikov menggambarkan Scrum sebagai "Proses pengembangan perangkat lunak *Agile* yang dirancang untuk

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

Scrum menawarkan cara yang dapat disesuaikan untuk bekerja pada proyek yang berbeda yang memiliki berbagai persyaratan dan Scrum memiliki keunggulan seperti pemilihan persyaratan atau spesifikasi kebutuhan yang fleksibel untuk *sprint* dan tidak ada prosedur khusus yang harus diikuti. Prinsipnya adalah bekerja secara iteratif hingga mencapai waktu yang ditentukan dan dapat memenuhi kebutuhan konsumen.

M. Tim Scrum

Tim Scrum setidaknya terdiri dari *Product Owner*, *Development Team* dan *Scrum Master*. Tim Scrum adalah tim yang lintas fungsi, di mana anggotanya memiliki semua keterampilan yang diperlukan untuk melakukan pekerjaan tanpa bergantung pada orang lain di luar tim pada setiap *Sprint*. Anggota tim Scrum juga mengatur diri mereka sendiri yang berarti secara internal mereka memutuskan siapa melakukan apa, kapan, dan bagaimana. Berikut merupakan anggota-anggota dari Tim Scrum.

1. *Product Owner*

   Pemilik produk (*product owner*) adalah peran yang berfokus pada keberhasilan produk. Pemilik produk mencoba memaksimalkan nilai produk

perangkat lunak bagi penggunanya. Oleh karena itu, pemilik produk bekerja secara kolaboratif dengan tim pengembangan dalam satu tim dan memberikan arahan pengembangan produk melalui visi produk mereka. Peran ini mengutamakan pekerjaan tim pengembang agar dapat diperoleh nilai terbaik dari produk perangkat lunak secara bertahap dan iteratif.

Selain berkolaborasi dengan tim pengembangan, Pemilik produk membangun kemitraan aktif dengan para pemangku kepentingan. Pemangku kepentingan adalah siapa pun di luar tim yang tertarik pada keberhasilan produk perangkat lunak. Manajemen, pelanggan, dan sponsor produk, kita dapat menyebutnya sebagai pemangku kepentingan.

Pemilik produk adalah individu, bukan kelompok. Pemilik produk mewakili suara para pemangku kepentingan, mencari kesamaan di antara banyak kepentingan produk melalui komunikasi yang baik.

2. *Development Team*

Tim pengembangan (*development team*) adalah tim dengan berbagai peran yang diperlukan untuk membangun produk perangkat lunak. Beberapa peran khusus yang sering kita temui adalah *software developer*, *UI/UX designer*, dan *quality assurance*. Peran-peran ini berkolaborasi setiap hari dalam tim pengembangan untuk membangun produk perangkat lunak sesuai dengan prioritas kerja yang disepakati dengan pemilik produk.

Di dalam Scrum, siapa pun yang termasuk dalam tim pengembangan disebut pengembang. Tidak ada peran khusus dalam tim pengembangan seperti yang didefinisikan oleh Scrum. Setiap orang adalah pengembang, dan siapa pun dapat melakukan apa saja untuk membangun perangkat lunak berkualitas tinggi. Membuat tim pengembangan lintas fungsi dan mengatur diri sendiri

bukanlah tugas yang mudah. Ada proses yang membutuhkan waktu dan kesabaran untuk meningkatkan tim untuk mencapai level itu.

3. *Scrum Master*

Scrum master adalah peran yang berfokus pada keberhasilan implementasi proses perangkat lunak yang dihasilkan dari metode Scrum. *Scrum master* memastikan aktivitas pengembangan dengan mengikuti proses Scrum. *Scrum master* membantu pemilik produk dan tim pengembang bekerja sama dalam kerangka proses Scrum.

Sebagai pelatih, *scrum master* melatih tim Scrum untuk menggunakan Scrum dengan tepat dalam lingkungan organisasi sehari-hari untuk mendapatkan manfaat terbaik dari Scrum. *Scrum master* membantu semua orang di tim Scrum memahami tujuan dan manfaat Scrum untuk pengembangan produk perangkat lunak yang sukses.

Sebagai fasilitator, *scrum master* menjadi fasilitator di tim Scrum selama pertemuan Scrum. *Scrum master* mencari keputusan antara Pemilik Produk dan Tim Pengembang untuk memberikan perangkat lunak terbaik yang dihasilkan melalui musyawarah.

Sebagai agen perubahan, *scrum master* membawa perubahan pada tim dan organisasi menuju implementasi proses Scrum yang lebih baik. Untuk organisasi baru, *scrum master* memperkenalkan organisasi ke Scrum dan membantu organisasi mengadopsi Scrum. *Scrum master* memandu proses perubahan dari proses perangkat lunak yang tidak terdefinisi ke proses Scrum.

N. Aktivitas-aktivitas Scrum (*Scrum Events*)

Aktivitas-aktivitas pada Scrum dibuat untuk menciptakan kontinuitas dan mengurangi fase lain yang tidak terdaftar di Scrum. Terjadinya kegagalan dalam menerapkan salah satu dari tahapan ini akan mengurangi transparansi dan menghilangkan peluang untuk peninjauan dan perubahan. Aktivitas-aktivitas yang terdapat pada Scrum adalah sebagai berikut.

1. *Sprint*

   Sprint merupakan tahap pengembangan perangkat lunak dengan durasi maksimum satu bulan dan memiliki durasi yang konsisten selama proses pengembangan produk. *Sprint* baru akan langsung dimulai segera setelah *Sprint* sebelumnya selesai. Sprint terdiri dari *Sprint Planning*, *Daily Scrum*, *Sprint Review* dan *Sprint Retrospective*.

2. *Sprint Planning*

   Pekerjaan yang dilakukan dalam *Sprint* direncanakan dalam tahap *Sprint Planning*. Rencana tersebut dibuat oleh semua anggota dalam tim Scrum. Untuk *Sprint* yang berdurasi satu bulan, batas waktu *Sprint Planning* maksimal 8 jam.

3. *Daily Scrum*

   Daily Scrum adalah aktivitas yang berlangsung hingga 15 menit dan bertujuan agar tim pengembang dapat menyinkronkan pekerjaan mereka dan membuat rencana untuk 24 jam ke depan. Hal ini dilakukan dengan meninjau pekerjaan sejak tahap *Daily Scrum* terakhir dan memperkirakan apa yang dapat dilakukan sebelum melanjutkan ke *Daily Scrum* berikutnya.

4. *Sprint Review*

Sprint Review diadakan di akhir setiap *Sprint* untuk meninjau iterasi dan merevisi *Product Backlog* sesuai kebutuhan. Selama *Sprint Review*, Tim Scrum dan pemangku kepentingan berkolaborasi untuk membahas pekerjaan yang telah dilakukan dalam *Sprint* yang baru saja selesai. Berdasarkan hasil ini dan setiap perubahan pada *Product Backlog* selama Sprint, peserta berkolaborasi untuk menentukan apa yang dapat dilakukan di *Sprint* berikutnya untuk mengoptimalkan nilai produk. Pertemuan bersifat informal, bukan pertemuan status, dan pemaparan diharapkan dapat menghimpun masukan dan menumbuhkan semangat kerja sama.

5. *Sprint Retrospective*

   Sprint Retrospective adalah kesempatan bagi Tim Scrum untuk meninjau diri mereka masing-masing dan mengembangkan rencana untuk perbaikan di *Sprint* berikutnya. *Sprint Retrospective* dilakukan setelah *Sprint Review* selesai dan sebelum *Sprint Planning* berikutnya. Batas waktu maksimum tahap ini adalah 3 jam untuk *Sprint* yang berdurasi satu bulan. Untuk *Sprint* yang lebih pendek, batas waktunya biasanya lebih pendek. *Scrum Master* memastikan bahwa langkah-langkah ini dilakukan dan semua anggota memahami tujuan mereka. *Scrum Master* mendidik Tim Scrum untuk mengimplementasikannya dalam batas waktu yang telah ditentukan. *Scrum Master* berpartisipasi sebagai mitra yang bertanggung jawab atas setiap proses yang terjadi saat Scrum terjadi.

O. Komponen Scrum

Komponen Scrum mewakili pekerjaan atau nilai. Mereka dirancang untuk memaksimalkan transparansi informasi penting. Jadi setiap orang yang mengkajinya

memiliki dasar adaptasi yang sama. Berikut merupakan komponen-komponen yang terdapat pada metode Scrum.

1. *Product Backlog*

   Product Backlog adalah daftar apa yang dibutuhkan untuk meningkatkan produk. *Item* dari *Product Backlog* yang dapat dijalankan oleh Tim Scrum dalam sebuah *Sprint* dianggap siap untuk diseleksi dalam aktivitas *Sprint Planning*. Mereka biasanya mendapatkan transparansi ini setelah melakukan kegiatan revisi. Perbaikan *Product Backlog* adalah operasi yang memecah dan mendefinisikan lebih lanjut *item-item* Product Backlog menjadi *item* yang lebih kecil dan lebih tepat. Menambahkan detail seperti deskripsi, urutan, dan ukuran diperlukan. Pengembang yang akan melakukan pekerjaan bertanggung jawab atas pengukuran. Pemilik produk dapat memengaruhi pengembang dengan membantu mereka memahami dan memilih.

2. *Sprint Backlog*

   Sprint Backlog mencakup tujuan Sprint (mengapa), item *Product Backlog* yang dipilih untuk *Sprint* (apa), dan rencana yang layak untuk memberikan *Increment* (bagaimana). *Sprint Backlog* adalah rencana yang dibuat oleh pengembang. Ini adalah gambaran waktu nyata dari pekerjaan yang direncanakan pengembang selama *Sprint* untuk mencapai tujuan *Sprint*. Oleh karena itu, *Sprint Backlog* diperbarui sepanjang *Sprint* karena semakin banyak yang dipelajari. Ini harus cukup detail sehingga mereka dapat memeriksa seluruh kemajuan mereka di *Daily Scrum*.

3. *Increment*

   Peningkatan (*Increment*) adalah komponen yang dianggap sebagai batu loncatan untuk mencapai tujuan produk. Setiap peningkatan melengkapi

semua tambahan sebelumnya dan diverifikasi secara menyeluruh untuk memastikan semua tambahan berfungsi secara maksimal. Beberapa penambahan dapat dilakukan di *Sprint*. Total yang ditambahkan diberikan dalam diskusi *Sprint*. Namun, Peningkatan dapat dikirim ke pemangku kepentingan sebelum *Sprint* selesai. Diskusi *Sprint* tidak boleh dianggap sebagai referensi untuk evaluasi. (Schwaber dan Sutherland, 2020)

BAB III

METODOLOGI PENELITIAN

A. Deskripsi Sistem

Gambar 3.1: *Wireframe* Tampilan Sistem

Sistem yang akan dibuat pada penelitian ini adalah sistem *search engine* yang berfungsi untuk pencarian *website* berbasis *terminal* yang akan menerima *input* berupa *keyword* dari pengguna dan sistem akan menampilkan hasil *website* yang relevan sesuai dengan *keyword* tersebut.

Pembuatan *search engine* ini menggunakan beberapa algoritma dan metode sebagai pendukungnya, yaitu algoritma *modified similarity based crawling* yang terdapat pada komponen *crawler*, metode *TF-IDF* untuk memberikan bobot pada dokumen berdasarkan *keyword*, dan metode *Google PageRank* untuk perangkingan *website*. Selain itu, proses pengembangan sistem ini memakai metode pengembangan Scrum.

Gambar 3.2: *Flowchart* Sistem

Proses dari sistem *search engine* dimulai dengan mengumpulkan data ke dalam *database* sebanyak-banyaknya menggunakan *crawler*, lalu data tersebut akan diolah melalui *preprocessing data* dan disimpan ke dalam *database*. Setelah data sudah siap, maka sistem dimulai dengan meminta *input* berupa *keyword* dari pengguna lalu *keyword* tersebut akan diproses melalui *page ranking* dan *document ranking*. Setelah itu hasil pencarian akan muncul sesuai dengan *keyword* yang dimasukkan.

B. Desain Penelitian

Agar penelitian lebih terstruktur dan memudahkan penulis melakukan penelitian, maka dibutuhkan desain penelitian. Desain penelitian mencakup tahapan-tahapan yang dilakukan penulis dalam pembuatan sistem dengan menggunakan metode Scrum. Tahapan tersebut dapat dilihat pada gambar 3.3.

Gambar 3.3: Desain Penelitian

Tahap pertama yang dilakukan pada penelitian ini adalah mendefinisikan *product backlog* dari sistem yang akan dibuat, lalu membuat jadwal pengerjaan atau *sprint backlog*. Setelah itu, *sprint* dimulai dengan diselingi *daily scrum* untuk melaporkan *progress* dan menentukan *task* selanjutnya. Setelah sistem selesai dikerjakan di dalam *sprint*, maka dilakukan pengujian sistem.

C. Alat dan Bahan Penelitian

Dalam penelitian ini, beberapa alat berupa perangkat keras yang digunakan untuk menunjang pembuatan sistem adalah sebagai berikut:

1. Laptop yang mempunyai CPU Intel i7-6700hq dan RAM 20GB

2. LCD Monitor dengan resolusi 1920 x 1080 *pixel*
3. Koneksi berbasis *Wi-Fi*

Perangkat keras di atas sudah memenuhi persyaratan untuk menjalankan aplikasi *Python* sebagai *server* maupun sebagai *interpreter* dan dapat menjalankan *server* untuk *database MySQL*. Selain perangkat keras, perangkat lunak yang dipakai untuk membuat sistem ini adalah sebagai berikut:

1. Windows 10 64-bit Operating System
2. Visual Studio Code sebagai *code editor*
3. XAMPP untuk menjalankan *MySQL database*
4. Python 3 untuk menjalankan program Python

D. Perancangan Sistem

Perancangan sistem pada penelitian ini sesuai dengan komponen yang ada di dalam metode Scrum. Komponen-komponen tersebut terdiri dari *product backlog*, *sprint backlog*, *sprint*, *daily scrum*, dan *pengujian sistem*. Berikut merupakan rincian dari perancangan sistem sesuai dengan komponen Scrum.

1. *Product Backlog*

Berikut merupakan tabel *product backlog* sistem *search engine* yang telah dibuat berdasarkan diskusi dengan *Scrum Master*. Transkrip percakapan dengan *Scrum Master* terdapat pada **Lampiran A**. Setiap *item* dari *product backlog* ini akan diimplementasikan pada *sprint* dari awal hingga selesai.

Tabel 3.1: *Product Backlog*

No	***Story***	***Sprint***	***Status***	***Priority***
1	Mengubah struktur project dari program *crawler* menjadi modular	1	Completed	Penting
2	Konfigurasi program *crawler* sebagai *background service* di sistem operasi	2		Penting
3	Perancangan struktur *package*, *class*, dan fungsi program *TF IDF*	3		Penting
4	Perancangan struktur *package*, *class*, dan fungsi program *Page Rank*	4		Penting
5	Konfigurasi program *page rank* sebagai *background service* di sistem operasi	5		Penting
6	Merancang *REST API* untuk fungsi utama *TF IDF*	6		Penting
7	Perancangan struktur *project* arsitektur *search engine*	7		Penting
8	Perancangan struktur kode untuk modul pengindeks *Generalized Suffix Tree*	8		Penting

Berdasarkan tabel 3.1 di atas, *Product Backlog* pada penelitian ini terdiri dari 4 komponen, yaitu *story*, *sprint*, *status*, dan *priority*. *Story* menggambarkan pekerjaan besar yang nantinya akan dipecah lagi menjadi *task-task* yang kecil di dalam *Sprint*. Komponen *sprint* pada tabel ini menandakan *sprint* berapa *story* tersebut akan dilaksanakan. Komponen *status* menjelaskan apakah *story* tersebut sudah selesai atau belum. Sedangkan komponen *priority* menjelaskan skala prioritas dari *story* tersebut.

2. *Sprint Backlog*

Sprint backlog adalah daftar pekerjaan yang perlu dikerjakan ataupun yang sudah dikerjakan pada *sprint*. Di dalam *sprint backlog*, berbagai *task* kecil dibuat berdasarkan *story* yang terdapat pada *product backlog*. Berikut merupakan rincian dari *sprint backlog* yang pertama.

1. *Sprint-1*

   Sprint-1 dilakukan sepekan pada tanggal 16 April 2022 sampai dengan 22 April 2022. *Story* pertama pada *product backlog* dipecah menjadi beberapa *task* sebagai berikut.

Tabel 3.2: *Sprint 1 Backlog*

No	***Story***	***Task***	***Status***
1	Mengubah struktur kode dari program *crawler* menjadi modular	Membuat rancangan class diagram	Completed
2		Membuat rancangan entity relationship diagram untuk database	Completed
3		Mengubah struktur kode crawler sesuai dengan rancangan class diagram	Completed
4		Implementasi threading untuk menjalankan lebih dari satu proses dalam satu waktu	Completed
5		Testing program versi terbaru	Completed

3. *Daily Scrum*, *Sprint Review*, dan *Sprint Retrospective*

Dikarenakan sumber daya dan waktu yang terbatas, aktivitas *daily scrum* akan digantikan dengan cara mencatat hal-hal penting dan hambatan yang terjadi setiap hari sebagai *note* di *Github Projects*.

Sprint review, dan *sprint retrospective* dilakukan pada awal pekan yaitu di hari Selasa melalui *voice call*. Pada awal pekan ini akan dilakukan evaluasi mengenai perkembangan sistem maupun hambatan yang terjadi selama pengerjaan di setiap *sprint*.

4. *Sprint 1 Report*

Setelah merencanakan *sprint backlog*, pekerjaan pada *sprint* ini harus mengikuti rencana kerja yang ditentukan oleh *sprint backlog*. Tujuan dari *sprint-1* ini adalah mengubah struktur kode program *crawler* sebelumnya menjadi modular. Untuk manajemen *task* dan *progress*, penulis menggunakan *GitHub Projects* seperti pada gambar 3.4.

Gambar 3.4: *Github Projects Sprint-1*

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

1. *Entity Relationship Diagram Database*

   Entity Relationship Diagram (ERD) pada *sprint-1* ini menggambarkan masing-masing entitas dan relasi antar entitas pada *database* untuk *crawler*.

Gambar 3.5: *Entity Relationship Diagram Sprint-1*

Tabel-tabel yang dibutuhkan untuk menyimpan data pada *crawler* yaitu tabel *crawling*, *page_linking*, *page_information*, *page_images*, *page_tables*, *page_styles*, *page_forms*, *page_list*, dan *page_scripts*.

2. *Class Diagram*

   Class diagram pada gambar 3.6 menggambarkan kelas-kelas dalam sistem yang dipakai *crawler*. Rancangan tersebut merupakan rancangan *class diagram* dari program *crawler* yang modular.

Gambar 3.6: *Class Diagram Sprint-1*

Pada rancangan *class diagram crawler* ini, terdapat 4 kelas yang akan dibuat, yaitu kelas *database* untuk koneksi dan pembuatan *table* pada *database*, kelas *page content* yang berisi fungsi-fungsi untuk menyimpan konten dari halaman *website*, kelas *util* yang terdapat fungsi-fungsi pendukung, dan kelas *crawl* sebagai kelas utama untuk menjalankan *crawler*.

3. *Source code*

   Source code dari *sprint-1* terdapat di *Github Repository*, yang dapat diakses pada link https://github.com/lazuardyk/search-engine/tree/sprint-1

4. Pengujian Sistem *Crawler*

Pengujian sistem dilakukan dengan menjalankan program *crawler* yang dilakukan penelitian sebelumnya dan versi yang terbaru melalui *terminal command prompt* dan menargetkan situs https://www.indosport.com/ dengan *keyword* berupa "barcelona". Kedua program tersebut dijalankan selama 2 jam dan pada versi crawler terbaru menggunakan 3 *threads* karena menerapkan konsep *multithreading*.

```
Administrator: cmd - python
C:\Users\lazua\Documents\crawler>python modified_crawling.py
URL Awal: https://www.indosport.com/
Keyword: barcelona
page yang sedang di crawl: https://www.indosport.com/
page yang sedang di crawl: https://www.indosport.com
page yang sedang di crawl: https://www.indosport.com/sepakbola
page yang sedang di crawl: https://www.indosport.com/liga-indonesia
page yang sedang di crawl: https://www.indosport.com/liga-spanyol
page yang sedang di crawl: https://www.indosport.com/liga-italia
page yang sedang di crawl: https://www.indosport.com/liga-champions
page yang sedang di crawl: https://www.indosport.com/liga-europa
page yang sedang di crawl: https://www.indosport.com/liga-inggris
page yang sedang di crawl: https://www.indosport.com/sportainment
page yang sedang di crawl: https://www.indosport.com/seleb-sport
page yang sedang di crawl: https://www.indosport.com/lifestyle
page yang sedang di crawl: https://www.indosport.com/in-depth
page yang sedang di crawl: https://www.indosport.com/otomotif
page yang sedang di crawl: https://www.indosport.com/motogp
page yang sedang di crawl: https://www.indosport.com/formula1
page yang sedang di crawl: https://www.indosport.com/otomania
page yang sedang di crawl: https://www.indosport.com/multisport
page yang sedang di crawl: https://www.indosport.com/basket
page yang sedang di crawl: https://www.indosport.com/raket
page yang sedang di crawl: https://www.indosport.com/mma
```

Gambar 3.7: Pengujian *crawler* penelitian sebelumnya

```
Administrator: cmd - python

C:\Users\lazua\Documents\search-engine>python main.py
Masukkan url awal: https://www.indosport.com/
Masukkan keyword: barcelona
Masukkan durasi (dalam detik): 7200
Masukkan total thread yang ingin dipakai: 3
page yang sedang di crawl: https://www.indosport.com/ <Thread(Thread-1, started 13708)>
page yang sedang di crawl: https://www.indosport.com <Thread(Thread-1, started 13708)>
page yang sedang di crawl: https://www.indosport.com/sepakbola <Thread(Thread-1, started 13708)>
page yang sedang di crawl: https://www.indosport.com/liga-indonesia <Thread(Thread-2, started 10968)>
page yang sedang di crawl: https://www.indosport.com/liga-spanyol <Thread(Thread-1, started 13708)>
page yang sedang di crawl: https://www.indosport.com/liga-italia <Thread(Thread-2, started 10968)>
page yang sedang di crawl: https://www.indosport.com/liga-champions <Thread(Thread-1, started 13708)>
page yang sedang di crawl: https://www.indosport.com/liga-europa <Thread(Thread-2, started 10968)>
page yang sedang di crawl: https://www.indosport.com/liga-inggris <Thread(Thread-1, started 13708)>
page yang sedang di crawl: https://www.indosport.com/sportainment <Thread(Thread-3, started 9944)>
page yang sedang di crawl: https://www.indosport.com/seleb-sport <Thread(Thread-2, started 10968)>
page yang sedang di crawl: https://www.indosport.com/lifestyle <Thread(Thread-3, started 9944)>
page yang sedang di crawl: https://www.indosport.com/in-depth <Thread(Thread-1, started 13708)>
page yang sedang di crawl: https://www.indosport.com/otomotif <Thread(Thread-2, started 10968)>
page yang sedang di crawl: https://www.indosport.com/motogp <Thread(Thread-3, started 9944)>
page yang sedang di crawl: https://www.indosport.com/formula1 <Thread(Thread-1, started 13708)>
page yang sedang di crawl: https://www.indosport.com/otomania <Thread(Thread-2, started 10968)>
page yang sedang di crawl: https://www.indosport.com/multisport <Thread(Thread-3, started 9944)>
page yang sedang di crawl: https://www.indosport.com/basket <Thread(Thread-1, started 13708)>
page yang sedang di crawl: https://www.indosport.com/raket <Thread(Thread-2, started 10968)>
page yang sedang di crawl: https://www.indosport.com/mma <Thread(Thread-3, started 9944)>
page yang sedang di crawl: https://www.indosport.com/futsal <Thread(Thread-1, started 13708)>
```

Gambar 3.8: Pengujian *crawler* versi terbaru

Gambar 3.7 dan gambar 3.8 merupakan tampilan *crawler* versi lama dan terbaru yang berjalan setelah dimasukkan *input* dari *user*. Hasil dari kedua *crawler* yang telah selesai dijalankan ini terdapat pada gambar 3.9 dan gambar 3.10.

Gambar 3.9: Hasil pengujian *crawler* penelitian sebelumnya

Gambar 3.10: Hasil pengujian *crawler* versi terbaru

Untuk situs awal https://www.indosport.com/ hasil yang didapat pada *crawler* penelitian sebelumnya adalah 2638 halaman dengan durasi 7277 detik, sedangkan pada *crawler* versi terbaru adalah 5638 halaman dengan durasi 7346 detik. Dengan demikian, waktu yang dibutuhkan untuk *crawling* satu halaman adalah 2,7 detik pada *crawler* versi lama dan 1,3 detik pada *crawler*

terbaru yang saat ini dikembangkan.

Filters

Mengandung kata:

Tabel	Tindakan	Baris	Jenis	Penyortiran	Ukuran	Beban
crawling	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	1	InnoDB	utf8mb4_general_ci	16.0 KB	-
page_forms	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	15,090	InnoDB	utf8mb4_general_ci	8.5 MB	-
page_images	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	~127,381	InnoDB	utf8mb4_general_ci	82.6 MB	-
page_information	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	5,180	InnoDB	utf8mb4_general_ci	44.6 MB	-
page_linking	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	~727,055	InnoDB	utf8mb4_general_ci	163.7 MB	-
page_list	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	~517,910	InnoDB	utf8mb4_general_ci	232.7 MB	-
page_scripts	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	~224,220	InnoDB	utf8mb4_general_ci	337.6 MB	-
page_styles	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	329	InnoDB	utf8mb4_general_ci	1.5 MB	-
page_tables	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	239	InnoDB	utf8mb4_general_ci	1.5 MB	-
9 tabel	Jumlah	~1,617,405	InnoDB	utf8mb4_general_ci	872.7 MB	0 B

Gambar 3.11: Hasil pengujian *crawler* terbaru 2

Konten *website* yang diperoleh dari proses pengujian *crawling* pada *crawler* versi terbaru untuk setiap tabelnya adalah 15.090 baris pada *page_forms*, 127.381 baris pada *page_images*, 5.180 baris pada *page_information*, 727.055 baris pada *page_linking*, 517.910 baris pada *page_list*, 224.220 baris pada page_scripts, 329 baris pada *page_styles*, dan 239 baris pada *page_tables*.

LAMPIRAN A

Transkrip Percakapan

Hari: Sabtu

Tanggal: 16 April 2022

PL: Penulis

SM: Scrum Master

PL: Bagaimana cara kerja sistem search engine ini?

SM: Kita akan membuat search engine yang berjalan di terminal dan menerima keyword pengguna dan juga bisa diakses melalui API sehingga bisa berguna jika ingin diintegrasikan ke tampilan yang lain.

PL: Apakah sistem ini akan berjalan di lokal atau online?

SM: Untuk saat ini, sistem akan berjalan di lokal.

PL: Apa saja yang dibutuhkan di product backlog?

SM: Story yang pertama adalah mengubah struktur kode program crawler yang sudah ada menjadi modular, yang kedua mengintegrasikan document ranking TF IDF ke dalam arsitektur, yang ketiga mengintegrasikan PageRank ke dalam arsitektur.

PL: Bagaimana prioritas masing-masing story tersebut?

SM: Sama penting.

PL: Apakah ada story lainnya?

SM: Story keempat adalah konfigurasi program sebagai background service di sistem operasi, dan ke yang lima untuk merancang REST API untuk fungsi utamanya.

PL: Apa saja fungsi utama dari API nya?

SM: Fungsi utamanya adalah melihat peringkat website berdasarkan PageRank, mencari dokumen yang relevan, dan melihat peringkat website secara overall.

PL: Baik. Bahasa pemrograman apa yang akan dipakai pada sistem ini?

SM: Karena web crawler dan komponen yang lain memakai Python, maka kita akan memakai bahasa Python 3 juga.

DAFTAR PUSTAKA

Feldman, R. dan Sanger, J. (2007). The text mining handbook- advanced approaches in analyzing unstructured data.

Manning, C., Raghavan, P., dan Schutze, H. (2009). An introduction to information retrieval (online edition). *Cambridge: Cambridge University Press*.

Page, L., Brin, S., Motwani, R., dan Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Schwaber, K. dan Sutherland, J. (2020). *The Definitive Guide to Scrum: The Rules of the Game*. ScrumGuides.org.

Sutherland, J., Viktorov, A., Blount, J., dan Puntikov, N. (2007). Distributed scrum: Agile project management with outsourced development teams. In *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*, pages 274a–274a.

Xing, W. dan Ghorbani, A. (2004). Weighted pagerank algorithm. In *Proceedings. Second Annual Conference on Communication Networks and Services Research, 2004.*, pages 305–314.