

**PERANCANGAN *USER INTERFACE SEARCH ENGINE*
DENGAN ADMIN CONSOLE UNTUK MANAJEMEN DAN
VISUALISASI DATA HASIL PENGINDEKSAN SEARCH
ENGINE**

Skripsi

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



**Oleh:
Aldian Asmara
1313618032**

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

2024

LEMBAR PENGESAHAN

LEMBAR PENGESAHAN

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

Nama : Aldian Asmara
No. Registrasi : 1313618032
Jurusan : Ilmu Komputer
Judul : Perancangan *user interface search engine* dengan *admin console* untuk manajemen dan visualisasi data hasil pengindeksan *search engine*

Menyatakan bahwa skripsi ini telah siap diajukan untuk skripsi.

Menyetujui,

Dosen Pembimbing I

Muhammad Eka Suryana, M.Kom

NIP. 198512232012121002

Dosen Pembimbing II

Med Irzal, M. Kom

NIP. 197706152003121001

Mengetahui,

Koordinator program Studi Ilmu Komputer

Dr. Ria Arafiyah, M. Si

NIP. 197511212005012004

LEMBAR PERNYATAAN

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul **Perancangan User Interface Search Engine dan Admin Console Untuk Manajemen dan Visualisasi Data Hasil Pengindeksan** 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 yang disebutkan dalam teks skripsi ini, telah dicantumkan dalam Daftar Pustaka sesuai dengan norma, kaidah dan etika penulisan ilmiah.

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

Jakarta, 12 Juni 2023

Aldian Asmara

LEMBAR PERSEMPAHAN

Untuk Ibu, Bapak, Adik-Adik dan Diriku

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT, karena dengan rahmat dan karunia-Nya, penulis dapat menyelesaikan skripsi yang berjudul "**Perancangan User Interface Search Engine dengan Admin Console Untuk Manajemen dan Visualisasi Data Hasil Pengindeksan**". Keberhasilan dalam menyusun proposal skripsi ini tidak lepas dari bantuan berbagai pihak yang memberikan masukan guna sempurnanya proposal skripsi ini. Oleh karena itu, dengan kerendahan hati penulis mengucapkan banyak terima kasih kepada:

1. Yth. Para petinggi di lingkungan FMIPA Universitas Negeri Jakarta.
2. Yth. Ibu Dr. Ria Arafiyah, M. Si selaku Koordinator Program Studi Ilmu Komputer.
3. Yth. Bapak Muhammad Eka Suryana, M.Kom selaku Dosen Pembimbing I yang telah membimbing, mengarahkan, serta memberikan saran dan koreksi terhadap proposal skripsi ini.
4. Yth. Bapak Med Irzal, M.Kom selaku Dosen Pembimbing II yang telah membimbing, mengarahkan, serta memberikan saran dan koreksi terhadap proposal skripsi ini.

Penulis menyadari bahwa penyusunan proposal skripsi ini masih jauh dari sempurna. Oleh karenanya, kritik dan saran yang bersifat membangun akan sangat membantu. Akhir kata, penulis berharap tugas akhir ini bermanfaat bagi semua pihak khususnya penulis sendiri.

Jakarta, 12 Januari 2024

Aldian Asmara

ABSTRAK

ALDIAN ASMARA. Perancangan *User Interface Search Engine* dan Admin Console Untuk Manajemen dan Visualisasi Data Hasil Pengindeksan. Skripsi. Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Jakarta. 2023. Di bawah bimbingan Muhammad Eka Suryana, M.Kom dan Med Irzal, M.Kom.

Mesin pencari merupakan sebuah program komputer yang berfungsi untuk membantu pengguna dalam menemukan informasi dengan kata kunci tertentu. Pada proses berjalannya dibutuhkan sebuah aplikasi untuk mengatur sumber daya *search engine* yang ada. Penelitian ini memiliki tujuan untuk membuat sebuah *admin panel* untuk manajemen sumber daya *search engine*. Jenis penelitian ini adalah pengembangan atau *research and development*. Informasi pendukung untuk melakukan penelitian ini berasal dari studi literatur jurnal-jurnal terkait dan diskusi yang diadakan peneliti dengan *stakeholder*. Proses pengembangan yang digunakan dalam penelitian ini menggunakan metode *Scrum*. Aplikasi yang akan dikembangkan menggunakan teknologi *Python* dan *Javascript*. Hasil akhir dari penelitian ini adalah sebuah aplikasi *admin panel* untuk manajemen *search engine* yang telah dibuat.

Kata kunci: *search engine*, aplikasi, *scrum*

ABSTRACT

ALDIAN ASMARA. *Design of a Search Engine User Interface and Admin Console for Managing dan Visualizing Index Result Data.* Thesis. Faculty of Mathematics and Natural Sciences, State University of Jakarta. 2023. Supervised by Muhammad Eka Suryana, M.Kom and Med Irzal, M.Kom.

Search engine is a computer program that helps people find the information they need with certain keywords. In the running process, search engine needs a program to manage their resources. This research aims to create an admin panel for managing search engine resources. This research falls under the category of Research and Development. All information that was required for conducting this research came from literatures of related journals and discussions held by researchers and stakeholders. The development process of this application employs the Scrum methodology. The application was built in Python and Javascript. The final result of this research is an admin panel for managing search engine resources.

Keywords: *search engine, application, scrum*

DAFTAR ISI

DAFTAR ISI	x
DAFTAR GAMBAR	xiii
DAFTAR TABEL	xiv
I PENDAHULUAN	1
A. Latar Belakang Masalah	1
B. Rumusan Masalah	3
C. Pembatasan Masalah	3
D. Tujuan Penelitian	4
E. Manfaat Penelitian	4
II KAJIAN PUSTAKA	5
A. Search Engine	5
B. Agile	5
C. Scrum	6
D. Perancangan User Interface	7
1. Layout dan Spacing	7
2. Text	11
3. Warna	22
E. Javascript	32
F. Three JS	33
1. <i>Camera</i>	33
2. <i>Mesh</i>	34
3. <i>Scene Graph</i>	35
4. <i>Primitives</i>	35
5. <i>Drag Controls</i>	36
G. <i>Unit Testing</i>	36
H. <i>User Acceptance Testing</i>	37
I. Visualisasi Data	37
J. Participatory Design	39
K. Analisa Desain Database	40
L. Celery	42
M. Mongo DB	42
III METODOLOGI PENELITIAN	49
A. Deskripsi Penelitian	49
B. Pengumpulan Data	49
C. Analisa Kebutuhan	50
D. Perancangan Sistem Dengan Scrum	50

E.	Use Case Diagram	53
F.	Pengujian Sistem	53
1.	<i>Unit Testing</i>	53
2.	<i>User Acceptance Test</i>	53
IV HASIL DAN PEMBAHASAN		55
A.	<i>Sprint 1</i>	55
B.	<i>Sprint 2</i>	64
C.	<i>Sprint 3</i>	68
D.	<i>Sprint 4</i>	73
E.	<i>Sprint 5</i>	74
F.	<i>Sprint 6</i>	84
G.	<i>Sprint 7</i>	87
H.	<i>Sprint 8</i>	91
I.	<i>Sprint 9</i>	96
J.	<i>Sprint 10</i>	97
K.	<i>Sprint 11</i>	98
L.	<i>Sprint 13</i>	101
M.	Hasil Pengujian <i>User Acceptance Test</i>	103
V KESIMPULAN DAN SARAN		104
A.	Kesimpulan	104
B.	Saran	104
DAFTAR PUSTAKA		106

DAFTAR GAMBAR

Gambar 2.1	Alur kerja scrum	7
Gambar 2.2	Penentuan white spacing dengan cara <i>decremental</i>	8
Gambar 2.3	Tampilan dashboard dengan nilai <i>white spacing</i> yang kecil . .	9
Gambar 2.4	Komponen mengisi seluruh <i>white space</i> dari sebuah halaman	9
Gambar 2.5	Komponen mengisi sebagian <i>white space</i> dari sebuah halaman	10
Gambar 2.6	Rangkaian nilai skala	11
Gambar 2.7	Penggunaan ukuran font yang berlebihan dalam tampilan . . .	12
Gambar 2.8	Menentukan skala font menggunakan metode modular	13
Gambar 2.9	Skala metode buatan sendiri	14
Gambar 2.10	Perbandingan <i>font</i> untuk <i>headline</i> dan <i>body</i>	15
Gambar 2.11	Pengurutan berdasarkan popularitas <i>font</i>	15
Gambar 2.12	Memuat banyak tulisan dalam sebuah <i>layout</i> yang ada	16
Gambar 2.13	Tampilan paragraf untuk ukuran 45-75 karakter	17
Gambar 2.14	Teks dengan komponen yang lebar dalam satu layout	18
Gambar 2.15	Komponen kartu	19
Gambar 2.16	Komponen kartu dengan komponen aksi dan judul didekatkan	19
Gambar 2.17	<i>Baseline</i>	20
Gambar 2.18	Komponen aksi dan judul diselaraskan menurut <i>baseline</i> . .	20
Gambar 2.19	<i>Letter spacing</i>	21
Gambar 2.20	Open Sans dan Oswald	21
Gambar 2.21	Open Sans digunakan untuk <i>headline</i>	22
Gambar 2.22	Penggunaan warna aksen untuk informasi	22
Gambar 2.23	Penggunaan warna aksen untuk aksi destruktif	23
Gambar 2.24	Menentukan warna basis yang tepat menggunakan komponen <i>button</i>	23
Gambar 2.25	Shade tergelap dan terterang	24
Gambar 2.26	Nilai <i>shade</i> untuk basis warna	24
Gambar 2.27	Pengisian nilai shade untuk nilai 300 dan 700	24
Gambar 2.28	Pengisian nilai shade untuk nilai 200, 400, 600 dan 700 . .	24
Gambar 2.29	Penggunaan <i>hex</i>	25
Gambar 2.30	Penggunaan <i>hsl</i>	25
Gambar 2.31	<i>Hue</i>	26
Gambar 2.32	<i>Saturation</i>	26
Gambar 2.33	<i>Lightness</i>	27
Gambar 2.34	Kontras untuk teks ukuran normal dan besar	28
Gambar 2.35	Teks putih diatas tampilan berwarna	28
Gambar 2.36	Tampilan warna yang gelap dapat mencuri perhatian pengguna	29
Gambar 2.37	Pembalikan kontras antara teks dengan latar belakang . . .	29
Gambar 2.38	Warna saja tidak cukup dalam menyampaikan informasi . . .	30
Gambar 2.39	Penambahan ikon disamping warna untuk menyampaikan informasi	30

Gambar 2.40 Menggunakan warna berbeda untuk setiap <i>trend line</i>	31
Gambar 2.41 Menggunakan kontras yang berbeda untuk setiap <i>trend line</i> . .	31
Gambar 2.42 Frustum Camera ThreeJS	34
Gambar 2.43 Mesh ThreeJS	34
Gambar 2.44 Scene Graph ThreeJS	35
Gambar 2.45 Primitives ThreeJS	36
Gambar 2.46 Skema database yang digunakan dalam proses crawling	41
Gambar 2.47 Penggunaan storage masing masing tabel dalam database . .	42
Gambar 2.48 Arsitektur MongoDB	43
Gambar 2.49 Perbandingan performa pemasukan data pengguna untuk kedua database	45
Gambar 2.50 MySQL vs MongoDB Select	46
Gambar 2.51 MySQL vs MongoDB update	47
Gambar 2.52 MySQL vs MongoDB delete	48
Gambar 3.1 Tahapan penelitian	49
Gambar 3.2 Desain Penelitian	51
Gambar 3.3 <i>Use Case Diagram</i>	53
Gambar 4.1 Desain Penelitian	57
Gambar 4.2 Desain Logo ONE <i>Search Engine</i>	57
Gambar 4.3 Desain sistem warna pada logo ONE	58
Gambar 4.4 Pemberian nilai <i>white spacing</i> awal yang besar terhadap komponen <i>button</i>	59
Gambar 4.5 Pemberian nilai <i>white spacing</i> secara <i>decremental</i> terhadap komponen <i>button</i> sehingga <i>white spacing</i> yang diberikan terasa cocok	59
Gambar 4.6 Rancangan tampilan peta situs	60
Gambar 4.7 Rancangan tampilan peta login untuk <i>staff</i>	61
Gambar 4.8 Rancangan tampilan untuk menambahkan <i>staff</i>	61
Gambar 4.9 Rancangan tampilan mengubah profile bagi <i>staff</i>	62
Gambar 4.10 Rancangan tampilan pencarian	63
Gambar 4.11 Rancangan tampilan hasil pencarian	64
Gambar 4.12 Tampilan halaman pencarian <i>search engine</i>	65
Gambar 4.13 Tampilan halaman hasil pencarian <i>search engine</i>	66
Gambar 4.14 Tampilan struktur kode projek <i>search engine</i>	67
Gambar 4.15 Tampilan halaman <i>overview crawling search engine</i>	68
Gambar 4.16 Tampilan halaman <i>domains crawling search engine</i>	69
Gambar 4.17 Tampilan halaman hasil <i>crawling</i> halaman <i>web search engine</i>	70
Gambar 4.18 Tampilan halaman detail halaman <i>web search engine</i>	71
Gambar 4.19 Tampilan halaman status <i>crawling search engine</i>	72
Gambar 4.20 Tampilan halaman daftar domain <i>crawling search engine</i> . .	72
Gambar 4.21 Tampilan halaman peta situs	74
Gambar 4.22 Tampilan halaman <i>overview page ranking search engine</i> . .	76

Gambar 4.23 Tampilan halaman <i>overview document ranking search engine</i>	77
Gambar 4.24 Tampilan halaman <i>overview document ranking search logs</i>	78
Gambar 4.25 Tampilan halaman daftar kata <i>document ranking</i>	79
Gambar 4.26 Tampilan halaman detail kata <i>document ranking</i>	80
Gambar 4.27 Tampilan halaman <i>overview page ranking search engine</i>	81
Gambar 4.28 Tampilan halaman tambah <i>staff</i>	81
Gambar 4.29 Tampilan halaman daftar <i>staff</i>	82
Gambar 4.30 Tampilan halaman edit <i>profile</i>	82
Gambar 4.31 Tampilan halaman <i>login staff</i>	83
Gambar 4.32 <i>REST API crawling status</i>	85
Gambar 4.33 <i>REST API crawling metrics</i>	85
Gambar 4.34 <i>REST API crawling start</i>	86
Gambar 4.35 <i>REST API crawling stop</i>	86
Gambar 4.36 <i>REST API domains</i>	87
Gambar 4.37 <i>REST API status document ranking</i>	88
Gambar 4.38 <i>REST API start document ranking</i>	89
Gambar 4.39 <i>REST API stop document ranking</i>	89
Gambar 4.40 <i>REST API</i> kata paling banyak dicari	90
Gambar 4.41 <i>REST API words</i>	90
Gambar 4.42 <i>REST API search logs</i>	91
Gambar 4.43 <i>REST API</i> untuk mendapatkan halaman web	92
Gambar 4.44 <i>REST API</i> untuk peta situs	93
Gambar 4.45 <i>REST API</i> untuk halaman hasil pencarian	93
Gambar 4.46 <i>REST API</i> untuk login	94
Gambar 4.47 <i>REST API</i> untuk <i>status page ranking</i>	94
Gambar 4.48 <i>REST API</i> untuk <i>stop page ranking</i>	95
Gambar 4.49 <i>REST API</i> untuk <i>start page ranking</i>	95
Gambar 4.50 <i>REST API</i> untuk <i>page ranking metrics</i>	96
Gambar 4.51 Desain infrastruktur <i>search engine</i>	97
Gambar 4.52 Grafik penggunaan memori aplikasi	102
Gambar 4.53 Grafik penggunaan memori aplikasi dari <i>tracemalloc</i>	103

DAFTAR TABEL

Tabel 3.1	<i>Product Backlog</i>	52
Tabel 3.2	Format <i>User Acceptance Test</i>	54
Tabel 4.1	<i>Sprint 1 Backlog</i>	55
Tabel 4.2	<i>Sistem desain sizing dan spacing</i>	56
Tabel 4.3	<i>Sprint 2 Backlog</i>	64
Tabel 4.4	<i>Unit Testing</i> Tampilan Pencarian	67
Tabel 4.5	<i>Sprint 3 Backlog</i>	68
Tabel 4.6	<i>Unit Testing</i> Tampilan Crawling	72
Tabel 4.7	<i>Sprint 4 Backlog</i>	73
Tabel 4.8	<i>Unit Testing</i> Tampilan Page Ranking	74
Tabel 4.9	<i>Sprint 5 Backlog</i>	75
Tabel 4.10	<i>Unit Testing</i> Tampilan Staff	83
Tabel 4.11	<i>Unit Testing</i> Tampilan Page Ranking	83
Tabel 4.12	<i>Unit Testing</i> Tampilan Document Ranking	84
Tabel 4.13	<i>Sprint 3 Backlog</i>	84
Tabel 4.14	<i>Sprint 7 Backlog</i>	87
Tabel 4.15	<i>Sprint 8 Backlog</i>	91
Tabel 4.16	<i>Sprint 10 Backlog</i>	96
Tabel 4.17	<i>Sprint 11 Backlog</i>	97
Tabel 4.18	<i>Sprint 11 Backlog</i>	99
Tabel 4.19	<i>Sprint 13 Backlog</i>	102
Tabel 4.20	Hasil Pengujian <i>User Acceptance Test</i>	103

BAB I

PENDAHULUAN

A. Latar Belakang Masalah

Dengan bertambah banyaknya informasi yang berada di internet setiap harinya, tentu saja mencari informasi yang kita inginkan secara manual di Web sangatlah memakan waktu dan tidak efisien. Oleh karena itulah *search engine* atau mesin pencari hadir untuk menangani masalah tersebut. Mesin pencari atau search engine adalah program berbasis web yang dapat diakses di internet yang memiliki tujuan utama yaitu mencari yang informasi yang relevan dengan cepat terhadap query yang pengguna kirim. Mesin pencari atau search engine bekerja dengan cara mencocokan query dari pengguna kepada index yang search engine atau mesin pencari telah buat

Mesin pencari atau yang biasa disebut *Search Engine* merupakan sebuah program komputer yang berguna untuk membantu pengguna dalam mencari situs web berdasarkan permintaan pencarian pengguna. Mesin pencari sebenarnya tidak berbeda dengan *website* pada umumnya, hanya saja perannya lebih terfokus pada pengumpulan dan pengorganisasian berbagai informasi di internet sesuai dengan kebutuhan penggunanya. Selain untuk memudahkan pencarian, mesin pencari juga berguna untuk meningkatkan pengunjung sebuah situs web.

Kebanyakan *search engine* atau mesin pencari yang ada di pasaran seperti Google, Bing dan Yahoo menyimpan aktivitas pengguna mereka dalam bentuk sebuah riwayat pencarian. Riwayat pencarian memberikan wawasan mengenai bagaimana suatu *search engine* atau mesin pencari digunakan dan apa ketertarikan pengguna saat ini. Hal ini dibuat mungkin dikarenakan riwayat pencarian menyimpan apa saja yang pengguna cari pada *search engine* atau mesin pencari dalam jangka waktu tertentu. Data riwayat pencarian ini dapat digunakan lebih jauh lagi untuk mengerti lebih dalam tentang pengguna.

Pada penelitian yang berjudul “*Web Search Result Optimization by Mining the Search Engine Query Logs*”, sebuah metode diperkenalkan untuk mengoptimisasi hasil pencarian yang dimana metode yang diperkenalkan mempelajari dari riwayat pencarian dari penggunanya. Metode yang diperkenalkan ini memiliki tujuan untuk mengurangi waktu navigasi hasil pencarian oleh pengguna

dengan cara memprediksi kebutuhan informasi dari pengguna. Metode yang diusulkan dari penelitian adalah klasterisasi query berdasarkan query user dan feedback user, kemudian halaman yang dikunjungi oleh user yang membentuk pola sequential dalam setiap cluster dihasilkan dengan algoritma GSP (*Generalized Sequential Patterns*). Tujuan akhirnya adalah melakukan perankingan kembali berdasarkan data sekuensial yang telah dihasilkan sebelumnya. Hasil dari penelitian yang dilakukan menunjukkan hasil yang memuaskan dalam hal mengurangi ruang pencarian pengguna dan meningkatkan efektivitas search engine. Pendekatan yang dilakukan penelitian ini memerlukan setiap user memiliki perankingan dokumen yang berbeda dari user yang lain dikarenakan satu user dengan user lainnya pasti memiliki query pencarian dan tanggapan terhadap hasil pencarian yang berbeda juga.(Sharma dkk., 2010)

Namun ada juga *search engine* atau mesin pencari yang tidak menyimpan Riwayat pencarian penggunanya, seperti DuckDuckGo. DuckDuckGo merupakan *search engine* atau mesin pencari yang memiliki tujuan agar penggunanya dapat menjelajah internet tanpa mengkhawatirkan data personal mereka dimanfaatkan oleh perusahaan lain. DuckDuckGo menjanjikan layanan pencarian yang privat, anonim dan menawarkan *built-in tracker blocking* sehingga situs yang pengguna kunjungi akan kesulitan mengumpulkan informasi mengenai pengguna. DuckDuckGo menawarkan layanannya dalam *platform* perangkat *mobile* dan ekstensi *desktop*.

Visualisasi data adalah metode utama untuk membantu data mendapatkan interpretasi data dan juga menemukan nilainya. Data disajikan secara visual untuk menyampaikan interpretasi dasar mengenai apa yang data katakan tanpa adanya kesulitan (Ajagbe dkk., 2020). Saat ini, visualisasi data atau visualisasi informasi menjadi topik yang menarik dan menjadi bidang penelitian yang luas. (Caldarola dkk., 2016)

Visualisasi grafik merupakan cara untuk menampilkan informasi yang terstruktur sebagai diagram dari grafik dan jaringan. (Fadli dkk., 2020). Visualisasi *graph* dapat dilakukan dengan bantuan library open source diantaranya seperti GraphViz, D3.js, VivaGraph dan lain lain.(Hu dan Nöllenburg, 2018)

Dalam search engine, *user interface* atau tampilan merupakan hal yang penting mengingat search engine sering sekali digunakan dalam kehidupan sehari-hari bahkan menjadi bagian hidup dari seseorang. Pada umumnya, skenario dari penggunaan *user interface* atau tampilan dari *search engine* adalah sebagai berikut: Pengguna memiliki kata yang ingin dicari dan mengirimkannya kepada

mesin pencari atau *search engine* (1). *Search engine* merespon kata yang dikirimkan oleh user (2). *Search engine* akan mencari dokumen yang sesuai dengan query yang pengguna kirim dan menampilkannya ke tampilan *search engine* (3). Yang terakhir user menentukan apakah dokumen yang diterima user relevan atau tidak dengan yang diharapkan pengguna (4). (Alonso dan Baeza-Yates, 2000).

Pada penelitian "Perancangan *User Interface Search Engine* Dengan *Admin Console* Untuk Manajemen dan Visualisasi Data Hasil Pengindeksan *Search Engine*" (Khatulistiwa, 2022) telah dirancang arsitektur *serch engine* berbasis *console*. Pada penelitian ini terdapat hal penting yang kurang ialah tampilan untuk pengguna *search engine*. Pada *Search Engine* ini telah dilengkapi dengan fitur *Admin Console* yang berguna untuk menyediakan suatu cara untuk memanajemen *search engine* yang sudah dibuat. Pada penelitian (Khatulistiwa, 2022) telah dibuat aplikasi berbasis flask yang mendukung *REST API* untuk menyediakan cara yang mudah bagi tampilan untuk berkomunikasi dengan data yang telah disediakan *search engine*.

Penelitian ini akan merancang tampilan dari *search engine* dengan *admin console* untuk visualisasi dan manajemen hasil indeks dengan mengintegrasikan penelitian dari Khatulistiwa (2022) yang berfokus pada perancangan arsitektur *search engine* dengan mengintegrasikan *web crawler*, algoritma *page ranking* dan *document ranking*.

B. Rumusan Masalah

Dari uraian latar belakang di atas, perumusan masalah pada penelitian ini adalah "Bagaimana perancangan *user interface search engine* dengan *admin console* untuk manajemen dan visualisasi data hasil pengindeksan".

C. Pembatasan Masalah

Pembatasan masalah pada penelitian ini antara lain:

1. Penelitian ini menggunakan *search engine* yang telah dibuat oleh Khatulistiwa (2022).
2. Rancangan tampilan yang akan dibuat hanya berfokus untuk tampilan *desktop*.

D. Tujuan Penelitian

1. Membuat *user interface* dari *search engine* dengan *admin console* yang telah dibuat pada penelitian Khatulistiwa (2022).

E. Manfaat Penelitian

1. Bagi penulis

Memperluas pengetahuan tentang *search engine*, bagaimana cara memvisualisasikan data yang didapat, 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 rancang bangun aplikasi *search engine*.

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. Search Engine

Search Engine merupakan sebuah perangkat lunak yang dirancang untuk mencari informasi dalam *internet*. Dalam penggunaannya, pengguna memasukan kata kunci yang ingin pengguna cari dalam *search engine* dan *search engine* akan menampilkan daftar dokumen, suara, gambar, *video* dan lain lain yang relevan dengan kata kunci yang pengguna masukkan.

Search engine menggunakan sebuah perangkat lunak bernama *crawler* yang bertugas untuk memindai dan meng-index halaman *web* yang berada di *internet*. *Crawler* ini mengikuti tautan dari satu halaman ke halaman lainnya mengumpulkan informasi mengenai konten dan struktur *website*. Data yang berhasil dikumpulkan tersebut lalu disimpan dalam sebuah *database* yang membentuk *search engine* index.

Beberapa contoh *search engine* yang populer pada saat ini adalah Google, Yahoo, Bing, Baidu dan Yandex.

B. Agile

Pengembangan perangkat lunak menggunakan *agile* merupakan salah satu metode pengembangan perangkat lunak yang ada. Kata "*agile*" memiliki arti cepat, ringan dan bebas bergerak. Konsep pengembangan perangkat lunak menggunakan *agile* ditemukan oleh Kent Beck dan 16 koleganya dengan menyatakan *agile* adalah cara dalam membangun sebuah perangkat lunak dengan cara mengerjakannya dan membantu satu sama lain dalam membangunnya dalam satu waktu. Dalam pengembangan perangkat lunak menggunakan *agile*, interaksi dan personil adalah hal yang penting dibandingkan proses dan alat kerja. Perangkat lunak yang bekerja lebih penting dari dokumentasi yang lengkap, Kolaborasi antara klien lebih penting dari negosiasi kontrak dan responsif dalam perubahan adalah hal yang lebih penting dari mengikuti rencana yang telah dibuat. Spertim model lainnya, *agile* memiliki kelebihan dan tidak cocok untuk semua tipe projek. *Agile* membuat model proses yang toleran terhadap perubahan kebutuhan sehingga perubahan dapat dilakukan dengan cepat (Adi, 2015).

C. Scrum

Scrum dikembangkan oleh Jeff Sutherland di tahun 1993 dan memiliki tujuan untuk menjadi metode pengembangan dan manajemen yang mematuhi prinsip *Agile*. Fokus dalam metode ini adalah "strategi, sebuah metode pengembangan perangkat lunak holistik yang fleksibel dimana tim pengembang bekerja sebagai satu unit untuk mencapai tujuan utama yang sama".

Dalam pelaksanaannya, *scrum* menjadi tiga bagian yaitu *Product Owner*, *Scrum Master* dan *Team*. *Product Owner* merupakan seseorang yang bertanggung jawab untuk menentukan spesifikasi atau perangkat lunak yang ingin dibangun. *Product Owner* akan membuat kebutuhan yang akan diselesaikan oleh tim atau lebih dikenal sebagai *Product Backlog*. *Team* merupakan suatu entitas yang mengerjakan projek seperti analis bisnis, sistem analis, pengembang, penguji dan lain lain. *Team* merupakan entitas yang bertanggung jawab dalam menyelesaikan *Product Backlog* yang telah disediakan oleh *Product Owner*, yang dimana setiap anggota dari *Team* bertanggung jawab dalam setiap tugas dalam *Product Backlog* yang telah dibagikan. *Scrum Master* adalah seseorang yang akan memperkenalkan dan mengimplementasikan bagaimana *Scrum* bekerja pada *Team* dan memastikan semua orang dalam projek mengimplementasikan metode *Scrum* (Adi, 2015).

Pengerjaan projek dengan metode *Scrum* dimulai dengan penggambaran bagaimana sistem yang akan dibuat. Selanjutnya, *Project Owner* menggambarkan proses bisnis atau rencara kedalam *Product Backlog*. *Product Backlog* merupakan sekumpulan rencana yang harus diselesaikan oleh *Team*. Terdapat sebuah *sprint* dalam metode *scrum*. *Sprint* merupakan tujuan yang ingin dicapai dalam *sprint* selanjutnya. Setiap *sprint* dibulai dengan *Sprint Meeting Planning* yang merupakan aktivitas untuk menentukan *sprint* apa yang akan dilakukan pada *sprint* dilakukan selanjutnya. Setiap harinya, setiap anggota *Team* berkumpul dan berdiskusi mengenai apa yang telah dilakukan setelah *Daily Scrum Meeting* sebelumnya, masalah apa yang dihadapi, dan apa yang akan dikerjakan selanjutnya. Pertemuan ini direncanakan oleh *Scrum Master* dan setiap penghujung *sprint* akan dilakukan pertemuan untuk mendemonstrasikan apa saja yang telah dikerjakan (Adi, 2015).



Gambar 2.1: Alur kerja scrum

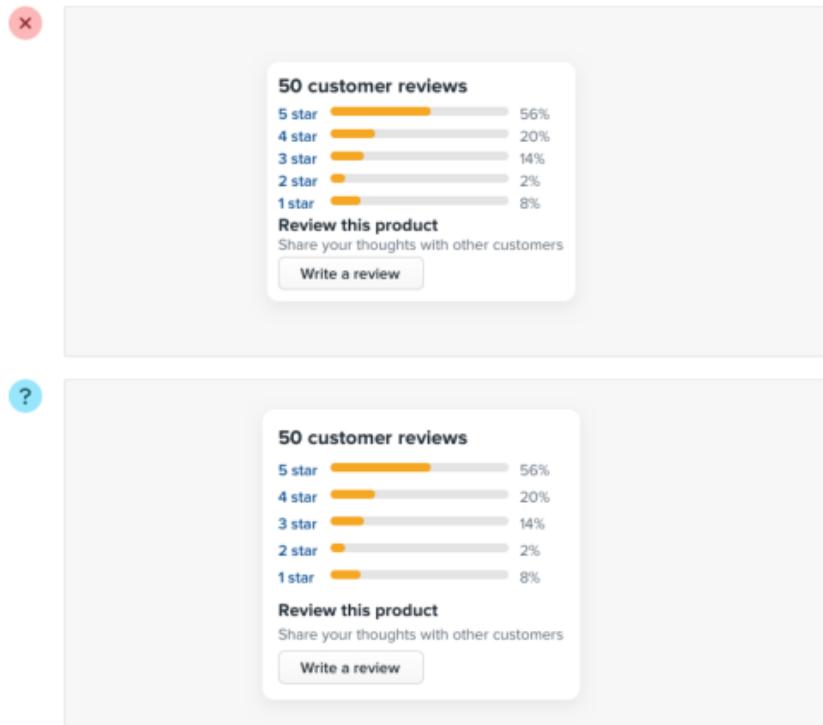
D. Perancangan User Interface

Menurut (M. Agus Muhyidin, 2020), *user interface* adalah ilmu tentang tata letak grafis suatu web atau aplikasi. Cakupan UI adalah tombol yang akan diklik oleh pengguna, teks, gambar, text entry fields, dan semua item yang berinteraksi dengan pengguna. Termasuk layout, animasi, transisi, dan semua interaksi kecil. UI mendesain semua elemen visual, bagaimana pengguna berinteraksi dengan halaman web dan apa yang ditampilkan di halaman web. Elemen visual yang ditangani oleh seorang desainer UI adalah skema warna, menentukan bentuk tombol, serta menentukan jenis font yang digunakan untuk teks. Desainer UI harus bisa membuat tampilan bagus yang akan meningkatkan kesetiaan pengguna.

1. Layout dan Spacing

White spacing merupakan jarak yang diberikan antara dua komponen baik itu jarak horizontal maupun vertikal. *White spacing* diperlukan untuk memberikan setiap komponen ruang untuk bernafas. Dalam membuat *white spacing* ada beberapa pendekatan diantaranya adalah pemberian *white space* secara *incremental* dari ukuran kecil hingga ke ukuran yang lebih besar. Pendekatan ini hanya meraih ukuran *white spacing* minimum untuk terlihat bagus dan pada dasarnya dalam tampilan yang baik biasanya diperlukan lebih banyak *white spacing*. Pendekatan yang lebih baik dalam perancangan *white spacing* yang baik adalah dengan mula mula memberikan komponen nilai *white spacing* yang besar, dan lakukan pengurangan nilai *white spacing* sampai pengguna merasa puas dengan nilai *white*

spacing yang diberikan.



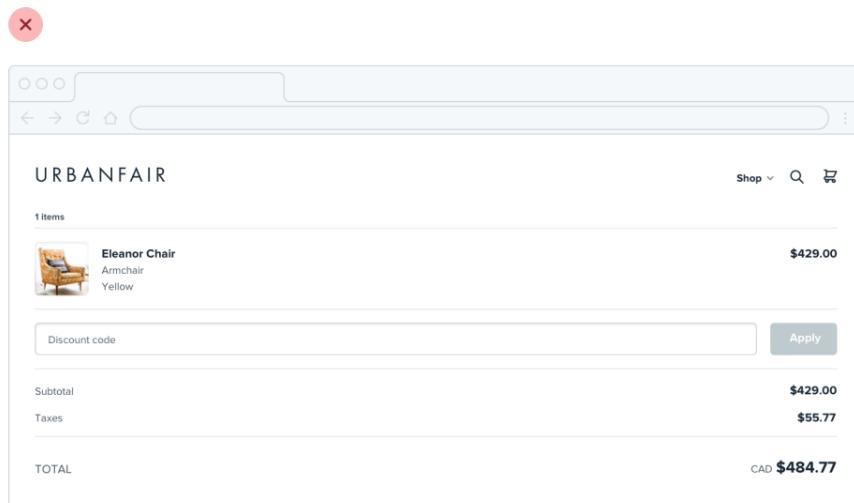
Gambar 2.2: Penentuan white spacing dengan cara *decremental*

Pada dasarnya komponen yang mempunyai banyak ruang untuk istirahat atau *white spacing* terlihat lebih bersih dan simpel, namun ada beberapa kasus dimana *white spacing* yang sedikit dan rapat dapat digunakan. Sebagai contoh, dalam perancangan tampilan *dashboard*, yang dimana dalam tampilan *dashboard* diperlukan untuk memuat informasi dalam jumlah banyak yang memungkinkan informasi tersebut terlihat dalam satu halaman.

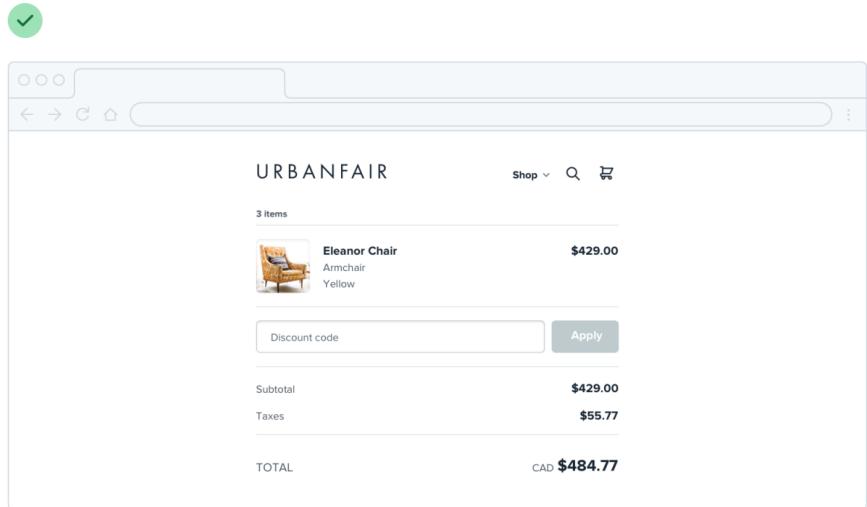
Game Summary																																																									
Canada	United States																																																								
TEAM STATS																																																									
SOG 30	FOS% 50%																																																								
PIM 6	HITS 36																																																								
	BLKS 12																																																								
SCORING																																																									
 Danial Berry Jason Chapman, Jake Sullivan 11:20 / 1st	<table border="1"> <thead> <tr> <th>#</th> <th>Forwards</th> <th>G</th> <th>A</th> <th>P</th> <th>+/-</th> <th>PIM</th> </tr> </thead> <tbody> <tr> <td>71</td> <td>W. Tran</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>15</td> <td>M. Hoffman</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>67</td> <td>T. Valdez</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>38</td> <td>H. Austin</td> <td>0</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>45</td> <td>D. Berry</td> <td>2</td> <td>1</td> <td>2</td> <td>+1</td> <td>2</td> </tr> <tr> <td>12</td> <td>J. Butler</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>19</td> <td>J. Chapman</td> <td>0</td> <td>1</td> <td>0</td> <td>-1</td> <td>0</td> </tr> </tbody> </table>	#	Forwards	G	A	P	+/-	PIM	71	W. Tran	0	0	0	0	0	15	M. Hoffman	0	0	0	0	0	67	T. Valdez	0	0	0	0	0	38	H. Austin	0	1	0	0	0	45	D. Berry	2	1	2	+1	2	12	J. Butler	0	0	0	0	0	19	J. Chapman	0	1	0	-1	0
#	Forwards	G	A	P	+/-	PIM																																																			
71	W. Tran	0	0	0	0	0																																																			
15	M. Hoffman	0	0	0	0	0																																																			
67	T. Valdez	0	0	0	0	0																																																			
38	H. Austin	0	1	0	0	0																																																			
45	D. Berry	2	1	2	+1	2																																																			
12	J. Butler	0	0	0	0	0																																																			
19	J. Chapman	0	1	0	-1	0																																																			

Gambar 2.3: Tampilan dashboard dengan nilai *white spacing* yang kecil

Dalam mendesain sebuah *layout* tidaklah harus mengisi seluruh *white space* yang ada. Mengisi seluruh *white space* yang ada hanya akan membuat tampilan lebih sulit untuk diinterpretasikan oleh pengguna.

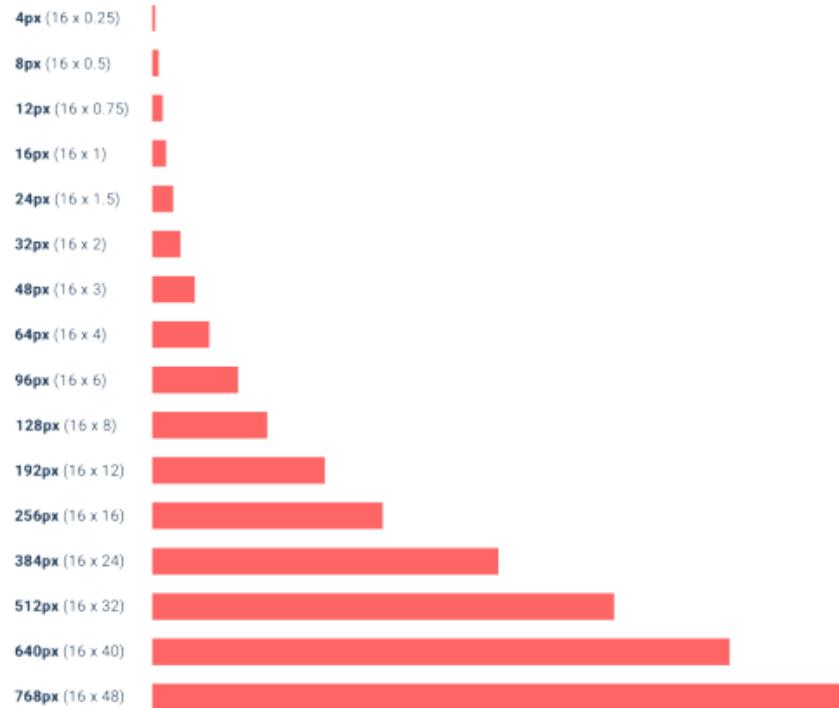


Gambar 2.4: Komponen mengisi seluruh *white space* dari sebuah halaman



Gambar 2.5: Komponen mengisi sebagian *white space* dari sebuah halaman

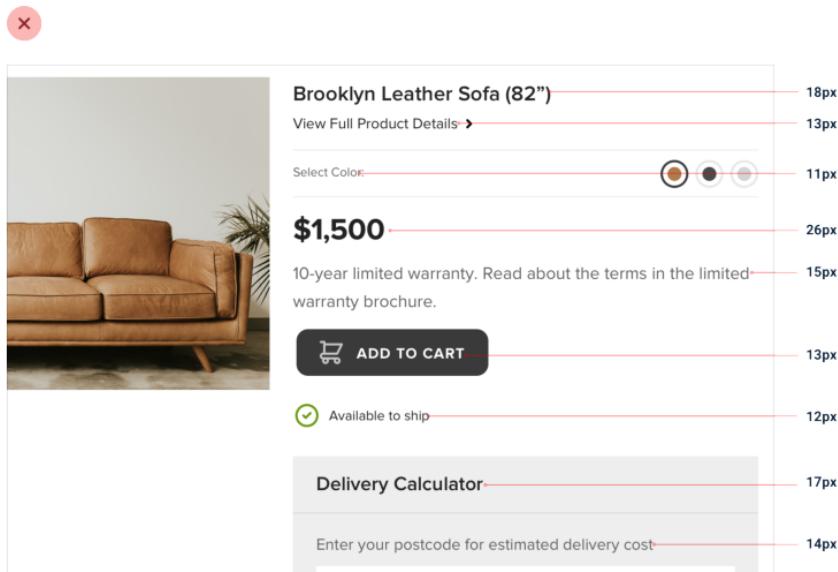
Penentuan sistem desain *spacing* dan *sizing* Dalam menentukan sebuah desain sistem ukuran *sizing* dan *spacing* tampilan, menghabiskan waktu dalam menentukan antara dua atau lebih nilai mana yang lebih baik untuk digunakan merupakan hal yang memakan waktu daripada menggunakan sistem yang telah ditentukan terlebih dahulu. Terdapat suatu pendekatan untuk menyelesaikan masalah tersebut dengan memulai dari menentukan nilai basis, dari nilai basis tersebut dapat dibuat rangkaian nilai skala dengan cara menggunakan nilai faktor dan mengalikannya dengan nilai basis yang telah ditentukan. Menggunakan nilai 16 *pixel* sebagai basisnya disarankan dikarenakan 16 *pixel* dapat dibagi secara baik dan merupakan setelan ukuran *font browser* awal. Berikut ini merupakan contoh dari pendekatan yang telah disebutkan:



Gambar 2.6: Rangkaian nilai skala

2. Text

Kebanyakan tampilan yang ada menggunakan banyak sekali ukuran *font*, bukan hal yang biasa untuk menemukan ukuran font dari ukuran 10 *pixel* sampai 24 *pixel* dalam satu tampilan. Hal ini merupakan hal yang tidak disarankan karena dua hal, desain yang tidak konsisten dan melambat pekerjaan.

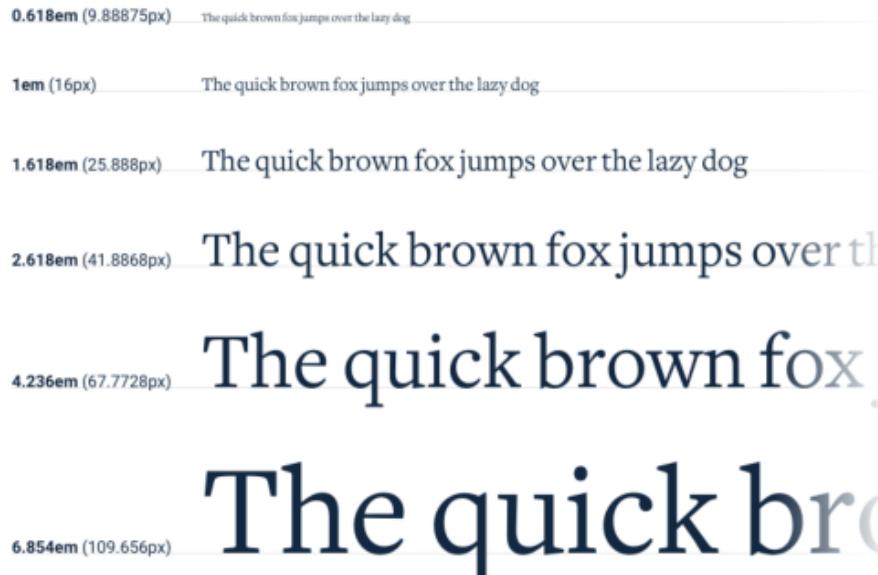


Gambar 2.7: Penggunaan ukuran font yang berlebihan dalam tampilan

Dalam membuat sistem desain font ada dua cara yaitu dengan menggunakan skala modular dan skala buatan sendiri.

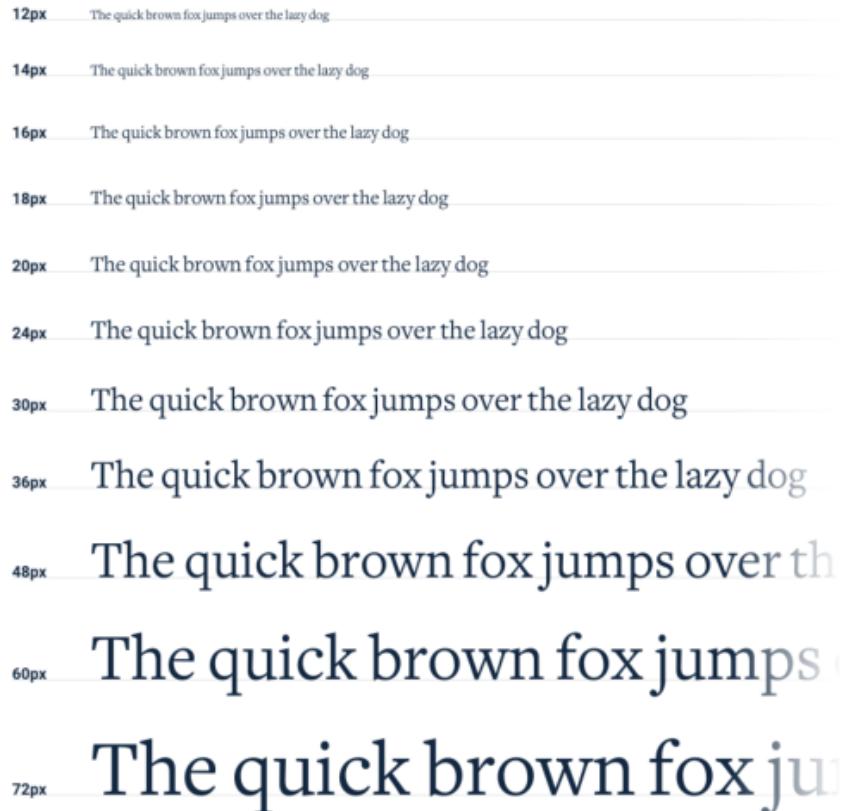
Dalam skala modular, digunakan rasio seperti 4:5 ("*a major third*"), 2:3 ("*perfect fifth*") dan 1:1.618 ("*golden ration*"). Dimulai dari menentukan nilai basis, mengaplikasikan rasio untuk mendapatkan angka selanjutnya dan mengaplikasikan nilai rasio tersebut lagi untuk mendapatkan nilai selanjutnya dan seterusnya. Pendekatan ini tampaknya menjanjikan, tapi dalam praktiknya, metode ini tidaklah sempurna karena beberapa alasan:

1. Dengan menggunakan skala modular dalam menentukan desain sistem font, ukuran skala akan berakhir menggunakan angka pecahan, seperti 31.25 *pixel*, 39.063 *pixel*, 48.828 *pixel* dan lain lainnya. Browser menangani subpixel sedikit berbeda sehingga nilai pecahan untuk ukuran text sebaiknya dihindari.
2. Dengan menggunakan metode ini, ukuran skala yang dihasilkan adalah seperti 12 *pixel*, 16 *pixel*, 21 *pixel* dan 28 *pixel*. Hal ini membuat pemilihan ukuran font menjadi terbatas, pada praktiknya biasanya dibutuhkan nilai antara 21 *pixel* dan 28 *pixel* atau nilai antara 12 *pixel* dan 16 *pixel*.



Gambar 2.8: Menentukan skala font menggunakan metode modular

Dalam skala buatan sendiri, penentuan nilai skala tidak dibatasi oleh formula matematika tetapi nilai skala ditentukan oleh pendesainnya sendiri. Dalam menggunakan metode ini tidak perlu lagi memerhatikan masalah seperti *subpixel rounding* pada *browser* dan juga dalam menggunakan metode ini, pendesain mempunyai kendali penuh dalam menentukan ukuran skala yang ada.

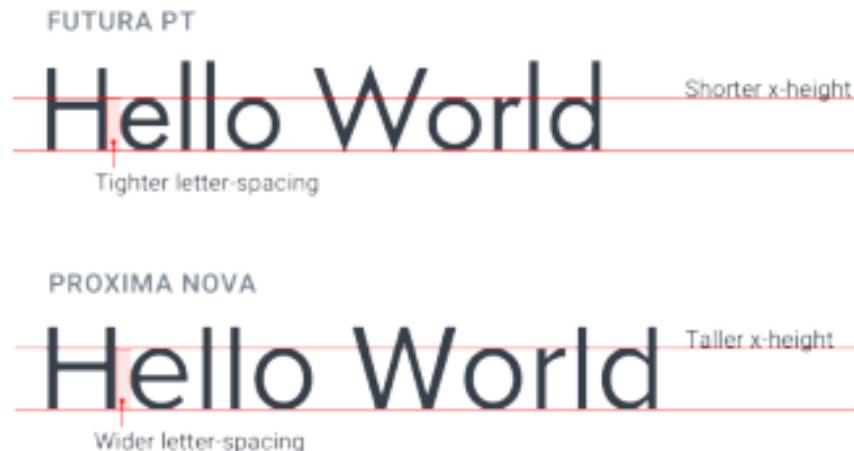


Gambar 2.9: Skala metode buatan sendiri

Pemilihan font yang akan digunakan dari ribuan font yang tersedia secara tepat adalah hal yang sangat memakan waktu, kemampuan untuk melihat dan menentukan font yang baik adalah kemampuan yang tidak dapat didapatkan secara singkat. Ada beberapa cara cepat untuk menentukan font yang baik guna mengurangi waktu yang diperlukan untuk mempercepat proses pemilihan font yaitu.

1. Untuk desain tampilan, pilihan yang aman adalah menggunakan *typeface sans-serif* seperti Helvetica. Jika ragu untuk menggunakan font yang dipilih, font bawaan dari perangkat pengguna adalah pilihan tepat. Penggunaan font dari bawaan perangkat pengguna membuat pengguna lebih familiar dengan tampilan yang dibuat karena pengguna sudah terbiasa dengan font yang digunakan di perangkat mereka.
2. Biasanya, font didesain untuk tujuan yang spesifik. Font yang memiliki *letter-spacing* yang lebih rapat dan huruf *lowercase* yang lebih pendek (*shorter x-height*) biasanya digunakan untuk *headline*, font ini tidak cocok

untuk digunakan sebagai font utama dari tampilan . Sementara untuk font yang digunakan untuk ukuran kecil seperti *body* biasanya mempunyai *letter-spacing* yang lebih lebar dan *lowercase* letter yang lebih tinggi (*taller x-height*).



Gambar 2.10: Perbandingan *font* untuk *headline* dan *body*

3. Font yang populer kemungkinan besar adalah font yang bagus. Beberapa penyedia font di internet menyediakan fitur untuk mengurutkan font menurut urutan popularitas nya sehingga dapat mengurangi jumlah font yang harus dipilih.

The screenshot shows a user interface for searching and sorting fonts. At the top, there is a dropdown menu labeled 'Sort by' with 'Popularity' selected, and a sample text area showing 'The quick brown fox jumps over the lazy c...'. Below this, three font families are listed in a grid format:

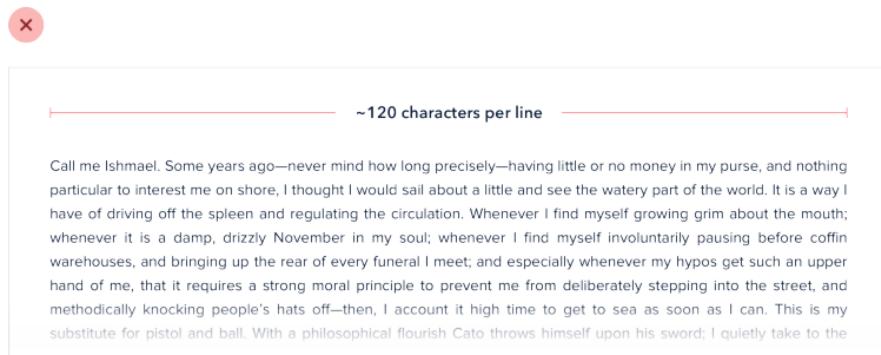
- EB Garamond** 10 Styles </>
- Alegreya** 10 Styles </>
- Cormorant Garamond** 10 Styles </>

Each listing includes the font name, the number of styles, a heart icon for favoriting, and a code icon (</>).

Gambar 2.11: Pengurutan berdasarkan popularitas *font*

4. Dimulai dengan mengunjungi beberapa situs favorit, biasanya dibalik situs tersebut, terdapat tim yang terdiri dari beberapa orang yang memiliki pengetahuan yang kuat tentang *typography*.

Ketika mendesain paragraf, mudah sekali untuk membuat kesalahan dengan memuat seluruh teks ke dalam *layout* yang ada dibandingkan dengan mencoba untuk membuat tampilan yang ramah untuk pembaca yang artinya teks akan terlihat sangat panjang dan sulit untuk dibaca.



Gambar 2.12: Memuat banyak tulisan dalam sebuah *layout* yang ada

Untuk pengalaman membaca yang baik, paragraf haruslah dibuat untuk dapat menampung 45 sampai 75 karakter. Cara yang mudah dalam *web* adalah menggunakan satuan unit *em*, yang dimana satuan unit ini relatif dengan ukuran font *web*. Ukuran yang tepat untuk ini adalah 20 *em* sampai 35 *em*.

45 - 55 characters per line

There was no possibility of taking a walk that day. We had been wandering, indeed, in the leafless shrubbery an hour in the morning; but since dinner (Mrs. Reed, when there was no company, dined early) the cold winter wind had brought with it clouds so sombre, and a rain so penetrating, that further out-door exercise was now out of the question.

55 - 65 characters per line

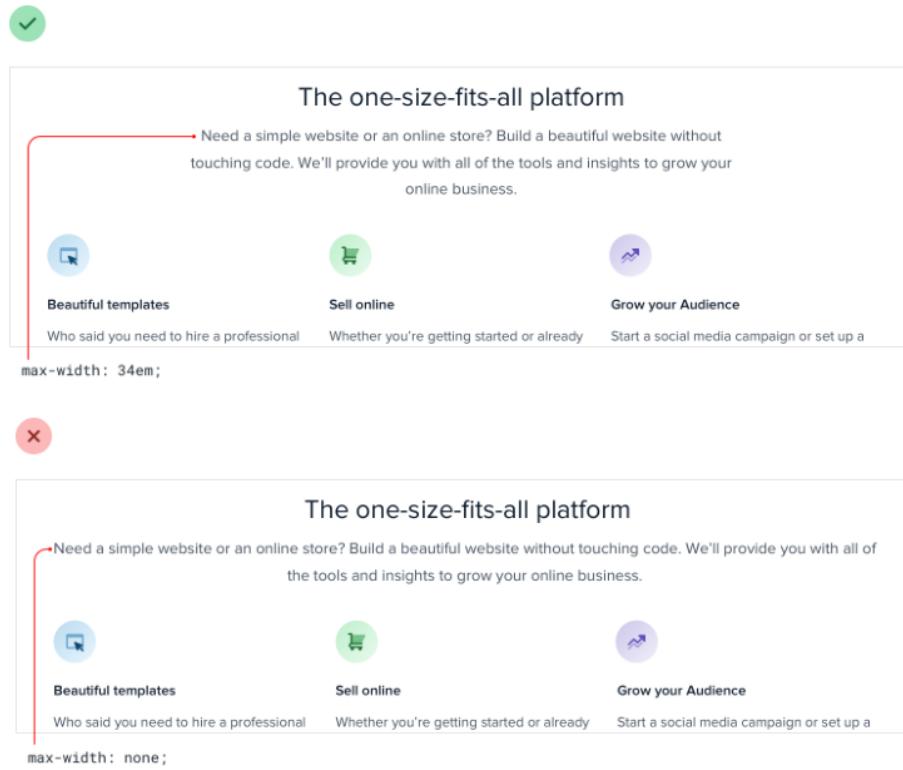
There was no possibility of taking a walk that day. We had been wandering, indeed, in the leafless shrubbery an hour in the morning; but since dinner (Mrs. Reed, when there was no company, dined early) the cold winter wind had brought with it clouds so sombre, and a rain so penetrating, that further out-door exercise was now out of the question.

65 - 75 characters per line

There was no possibility of taking a walk that day. We had been wandering, indeed, in the leafless shrubbery an hour in the morning; but since dinner (Mrs. Reed, when there was no company, dined early) the cold winter wind had brought with it clouds so sombre, and a rain so penetrating, that further out-door exercise was now out of the question.

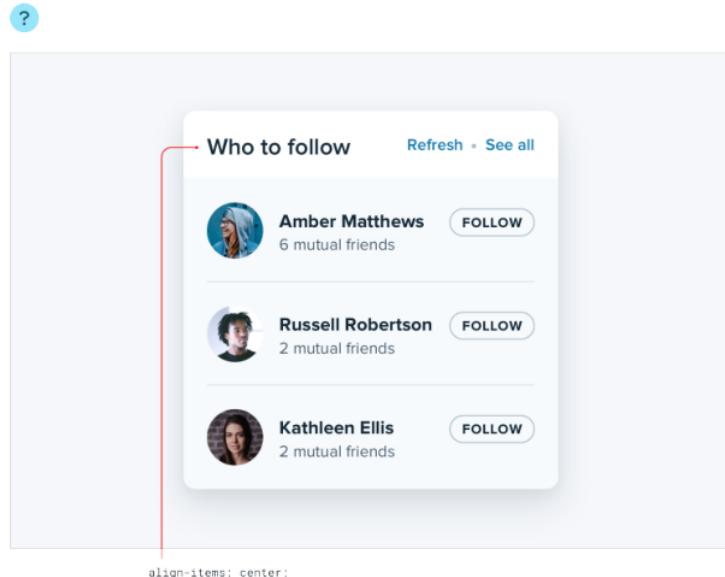
Gambar 2.13: Tampilan paragraf untuk ukuran 45-75 karakter

Jika paragraf mengandung konten seperti gambar atau komponen yang besar, lebar text paragraf haruslah tetap dibatasi meskipun keseluruhan konten paragraf harus melebihi dari batas untuk menampung keseluruhan komponen.



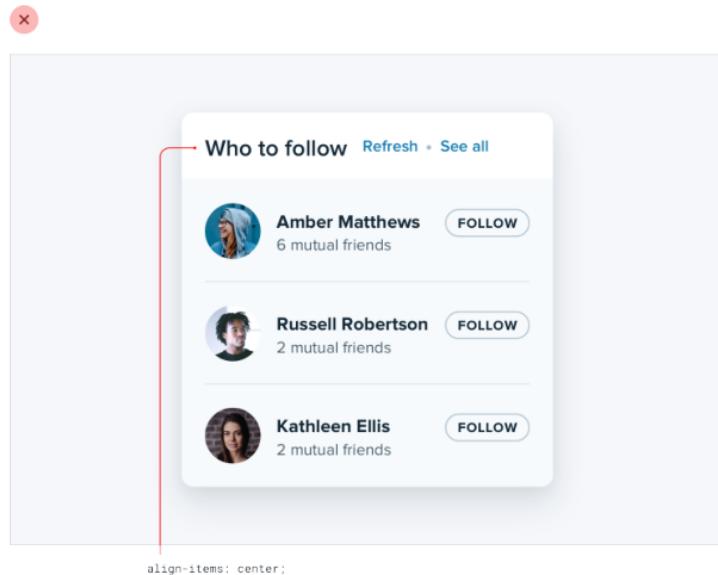
Gambar 2.14: Teks dengan komponen yang lebar dalam satu layout

Text Align Ada banyak situasi dimana diharuskan menggunakan beberapa ukuran *font* dalam satu baris. Sebagai contoh dalam mendesain kartu dimana judul kartu tersebut memerlukan ukuran font yang lebih besar dibandingkan dengan elemen di sampingnya. Saat mencampur ukuran font seperti ini, biasanya secara insting, desainer akan cenderung melakukan *center-ing* terhadap komponen tersebut untuk menciptakan keseimbangan.



Gambar 2.15: Komponen kartu

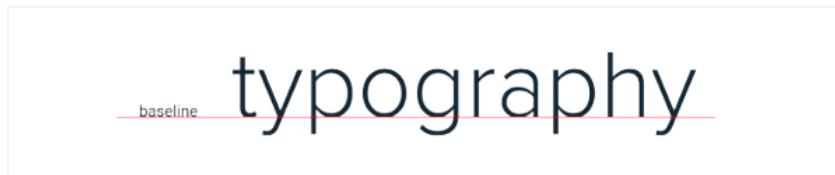
Ketika ada banyak ruang diantara dua komponen, hal ini terlihat baik baik saja. Namun jika kedua komponen teks tersebut didekatkan maka akan menjadi jelas bahwa tampilan akan terlihat buruk.



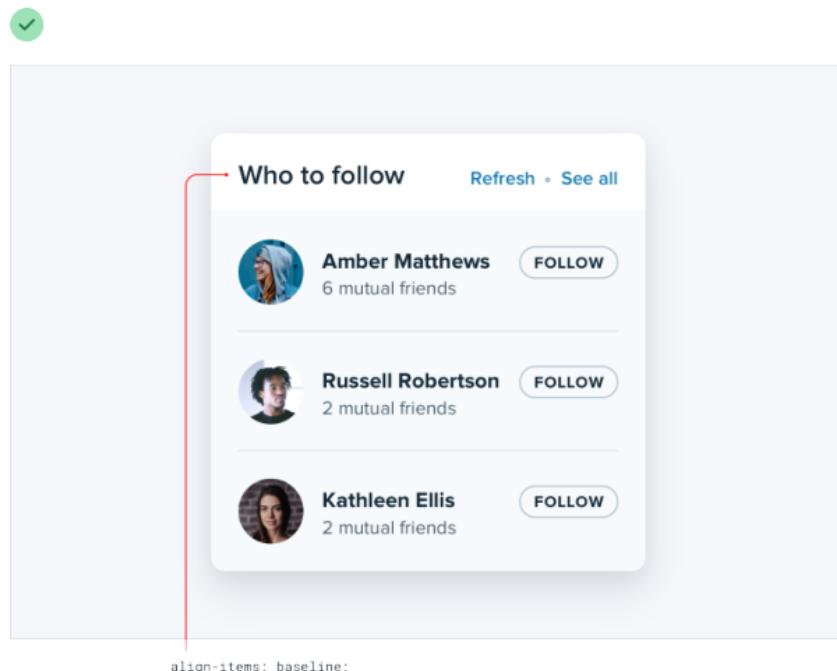
Gambar 2.16: Komponen kartu dengan komponen aksi dan judul didekatkan

Pendekatan terbaik adalah menyelaraskan kedua komponen tersebut dengan

baseline atau sebuah garis imajinari tempat teks berdiri yang digunakan oleh teks untuk berdiri.



Gambar 2.17: *Baseline*



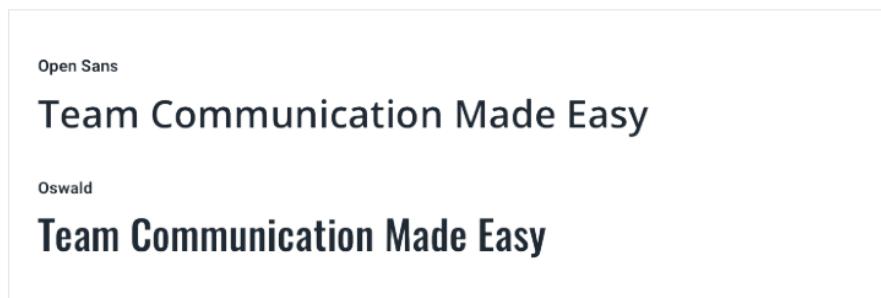
Gambar 2.18: Komponen aksi dan judul diselaraskan menurut *baseline*

Letter spacing Pada umumnya ada baiknya untuk mempercayakan *letter spacing* kepada pendesain *font* tersebut. Namun ada beberapa kasus yang dimana mengubah *letter spacing* dapat memperindah tampilan yang ada.

Tight letter-spacing letter-spacing: -0.05em;	There was no possibility of taking a walk that day. We had been wandering, indeed, in the leafless shrubbery an hour in the morning; but since dinner (Mrs. Reed, when there was no company, dined early) the cold winter wind had brought with it clouds so sombre, and a rain so penetrating, that further out-door exercise was now out of the question.
Normal letter-spacing letter-spacing: 0;	There was no possibility of taking a walk that day. We had been wandering, indeed, in the leafless shrubbery an hour in the morning; but since dinner (Mrs. Reed, when there was no company, dined early) the cold winter wind had brought with it clouds so sombre, and a rain so penetrating, that further out-door exercise was now out of the question.
Wide letter-spacing letter-spacing: 0.05em;	There was no possibility of taking a walk that day. We had been wandering, indeed, in the leafless shrubbery an hour in the morning; but since dinner (Mrs. Reed, when there was no company, dined early) the cold winter wind had brought with it clouds so sombre, and a rain so penetrating, that further out-door exercise was now out of the question.

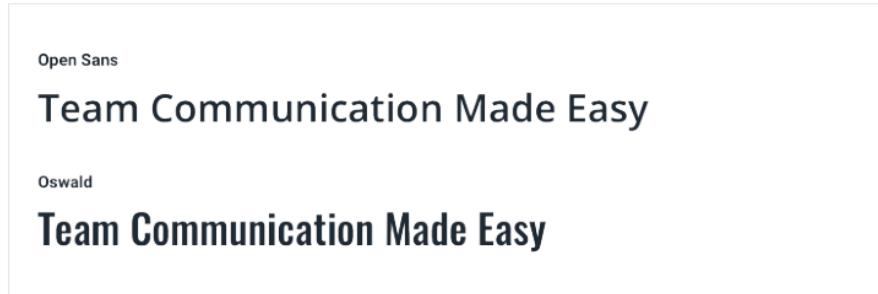
Gambar 2.19: *Letter spacing*

Ketika seseorang mendesain sebuah font, mereka mendesain *font* tersebut dengan tujuan tertentu. *Font family* seperti Open Sans didesain untuk keterbacaan dalam ukuran kecil yang dimana *letter spacing* dari font family tersebut terlihat lebih besar dibandingkan dengan *font family* seperti Oswald, yang dimana *font family* Oswald digunakan untuk kebutuhan seperti penulisan judul utama.



Gambar 2.20: Open Sans dan Oswald

Font yang dikhususkan untuk keterbacaan dalam ukuran kecil seperti Open Sans juga dapat digunakan untuk penulisan judul utama. Dengan mengurangi *letter spacing* untuk meniru fungsi dari *font family* seperti Oswald. Namun, hindari penggunaan sebaliknya, penggunaan *font family* judul utama untuk keterbacaan dalam ukuran kecil memiliki kemungkinan kecil untuk bekerja.



Gambar 2.21: Open Sans digunakan untuk *headline*

3. Warna

Color Palette Color palette dapat dibagi menjadi tiga kelompok yaitu

1. Abu abu

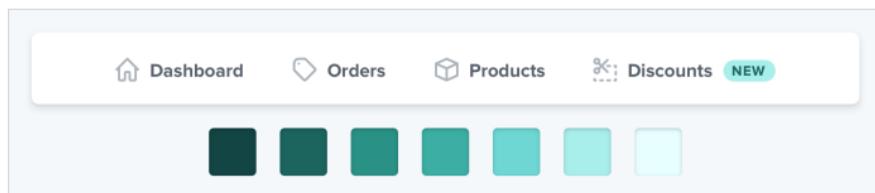
Warna yang biasanya terdapat pada beberapa komponen tampilan seperti *form*, panel, warna latar belakang dan teks.

2. Warna utama (primary color)

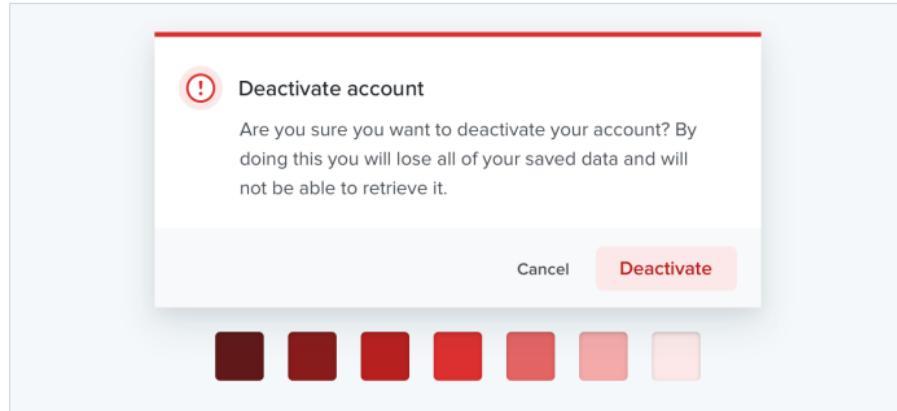
Warna yang digunakan dalam komponen seperti *button*, navigasi dan lain lain. Warna ini mendefinisikan bagaimana suatu website terlihat, seperti contohnya ketika memikirkan sebuah merek seperti Facebook maka akan terpikirkan warna biru yang merupakan ciri khas dari Facebook sendiri.

3. Warna aksen (accent color)

Warna aksen digunakan untuk menyampaikan maksud tertentu terhadap pengguna. Sebagai contoh, warna merah atau jingga digunakan untuk memikat pengguna terhadap fitur baru yang baru saja dirilis atau seperti warna merah yang digunakan untuk meminta konfirmasi pengguna untuk aksi yang destruktif.



Gambar 2.22: Penggunaan warna aksen untuk informasi



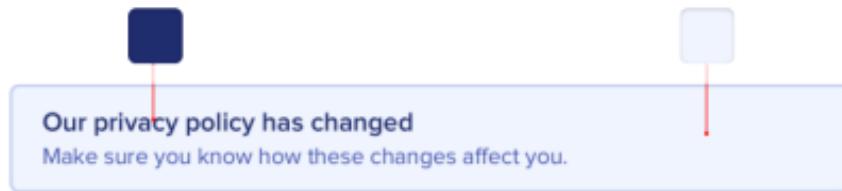
Gambar 2.23: Penggunaan warna aksen untuk aksi destruktif

Menentukan warna shade *Shade* dapat ditentukan dari warna basis yang ada. Warna basis merupakan warna yang berada di tengah tengah antara shade yang paling gelap dan shade yang paling terang. Dalam penentuan basis warna sendiri, tidak ada formula khusus, melainkan terdapat beberapa cara yang dapat dipakai dalam menentukan basis warna. Caranya adalah mengambil basis warna shade yang cocok digunakan untuk warna *background* dari elemen *button*.



Gambar 2.24: Menentukan warna basis yang tepat menggunakan komponen *button*

Selanjutnya adalah menentukan sisi paling gelap dan sisi paling terang, dalam menentukan sisi yang paling gelap dan sisi yang paling terang dari warna shade tidak ada formula khusus yang dapat digunakan. Biasanya, sisi yang paling gelap digunakan untuk sebuah teks sedangkan sisi yang paling terang digunakan untuk sebuah *background*. Dalam penentuannya dapat dimulai dengan menentukan basis warna lalu mengatur atribut *saturation* dan *lightness* hingga dirasa cocok.



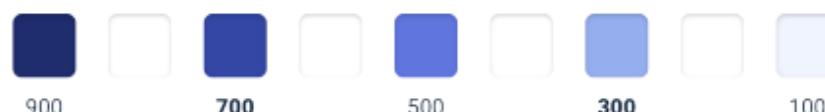
Gambar 2.25: Shade tergelap dan terterang

Saat selesai menentukan basis warna dan *shade* warna paling gelap dan paling terang, langkah selanjutnya adalah mengisi ruang kosong yang ada. Pada umumnya, dibutuhkan sekurang kurangnya 5 warna *shade* dalam suatu projek dan kurang lebih 10 warna *shade* jika tidak ingin merasa dibatasi dengan pilihan warna. Angka 9 adalah angka yang tepat dikarenakan mudah untuk dibagi dan membuat mengisi ruang kosong yang ada lebih mudah. 900 adalah warna *shade* paling gelap, 100 paling terang dan 500 adalah basis warna.



Gambar 2.26: Nilai *shade* untuk basis warna

Pengisian dimulai dari angka 700 dan 300 karena angka inilah yang berada di tengah tengah ruang kosong terus lanjutkan hingga semua ruang kosong terisi.

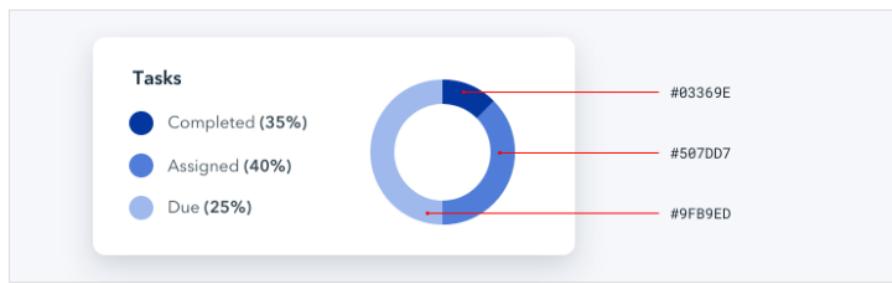


Gambar 2.27: Pengisian nilai shade untuk nilai 300 dan 700

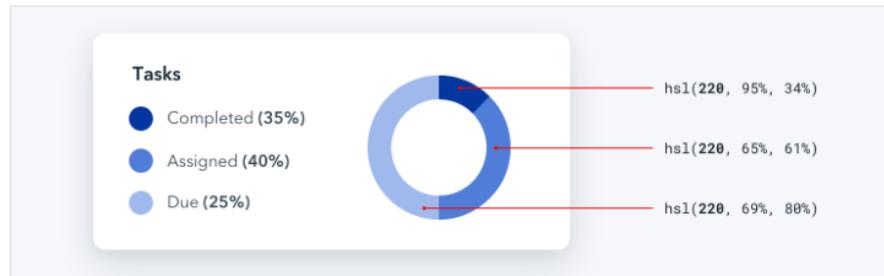


Gambar 2.28: Pengisian nilai shade untuk nilai 200, 400, 600 dan 700

HSL Hex dan RGB adalah dua format warna yang umum digunakan pada tampilan. Warna-warna hex dapat memiliki tampilan warna yang terlihat sama namun jika dilihat dari representasi kodennya, mereka terlihat tidak sama. HSL menyelesaikan masalah ini dengan mempresentasikan warna dengan atribut yang orang-orang dapat merasakannya secara intuitif. Atribut yang dimaksud adalah *hue*, *saturation* dan *lightness*.

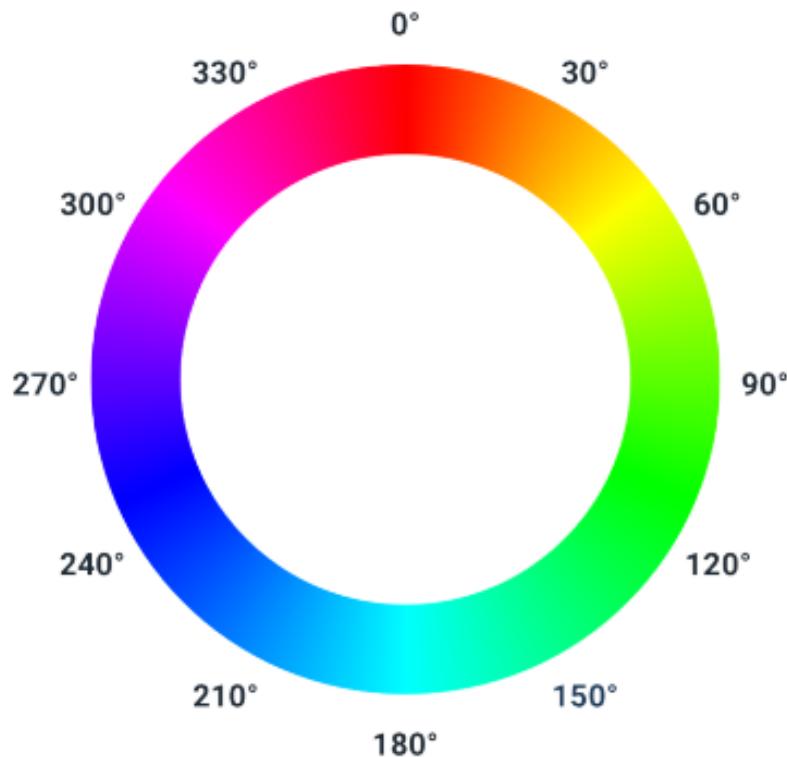


Gambar 2.29: Penggunaan *hex*



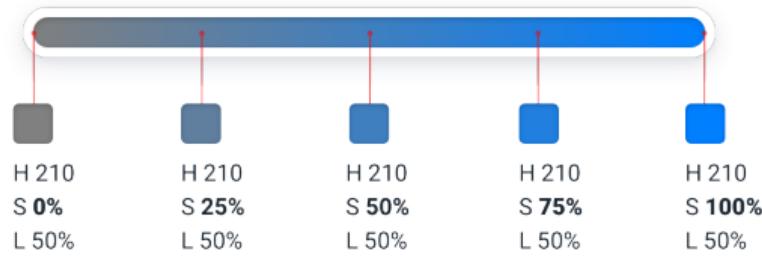
Gambar 2.30: Penggunaan *hsl*

Hue merupakan atribut yang merupakan posisi warna di *color wheel*. *Hue* diukur dalam satuan derajat yang dimana 0 derajat melambangkan merah, 120 derajat melambangkan hijau dan 240 derajat melambangkan biru.



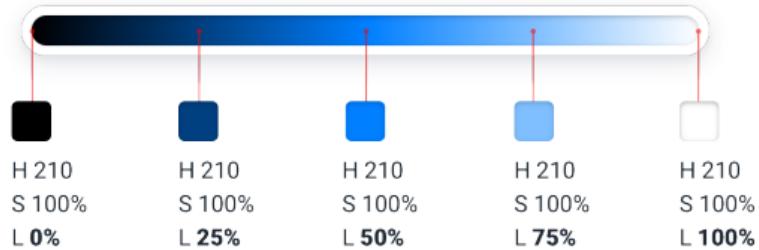
Gambar 2.31: Hue

Saturation adalah nilai yang mengukur seberapa mencolok atau jelas suatu warna, 0% *saturation* menandakan warna abu abu (tidak ada warna), sedangkan *saturation* 100% menandakan warna yang mencolok dan jelas. Tanpa *saturation* nilai *hue* tidaklah bermakna seberapapun nilainya karena warna akan tetap menjadi abu abu (tidak ada warna).



Gambar 2.32: Saturation

Atribut *lightness* merupakan nilai yang mengukur seberapa dekat atau jauhnya sebuah warna dengan warna hitam maupun putih. 0% *lightness* adalah warna hitam, 100% *lightness* merupakan warna putih dan 50% *lightness* merupakan warna asli dalam nilai *hue* tersebut.



Gambar 2.33: *Lightness*

Accessibility Untuk mendesain tampilan yang *accessible*, *Web Content Accessibility Guidelines (WCAG)* merekomendasikan teks normal yang memiliki ukuran dibawah 18 *pixel* memiliki kontras dengan perbandingan 4.5:1 dan teks yang lebih besar dari tersebut memiliki kontras tampilan setidaknya 3:1 untuk mendapat nilai kontras minimum (AA) dan kontras 7:1 untuk teks normal dibawah 18 dan 4.5:1 untuk teks yang lebih besar untuk mendapatkan nilai kontras yang tinggi (AAA).

Normal Text

EXAMPLE	COLOR	CONTRAST	GRADE
The five boxing wizards jump quickly.	hsl(0, 0%, 54%)	3.45:1	Fail
The five boxing wizards jump quickly.	hsl(0, 0%, 42%)	5.41:1	AA
The five boxing wizards jump quickly.	hsl(0, 0%, 33%)	7.57:1	AAA

Large Text

EXAMPLE	COLOR	CONTRAST	GRADE
The five boxing wizards jump...	hsl(0, 0%, 59%)	2.96:1	Fail
The five boxing wizards jump...	hsl(0, 0%, 54%)	3.45:1	AA
The five boxing wizards jump...	hsl(0, 0%, 42%)	5.41:1	AAA

Gambar 2.34: Kontras untuk teks ukuran normal dan besar

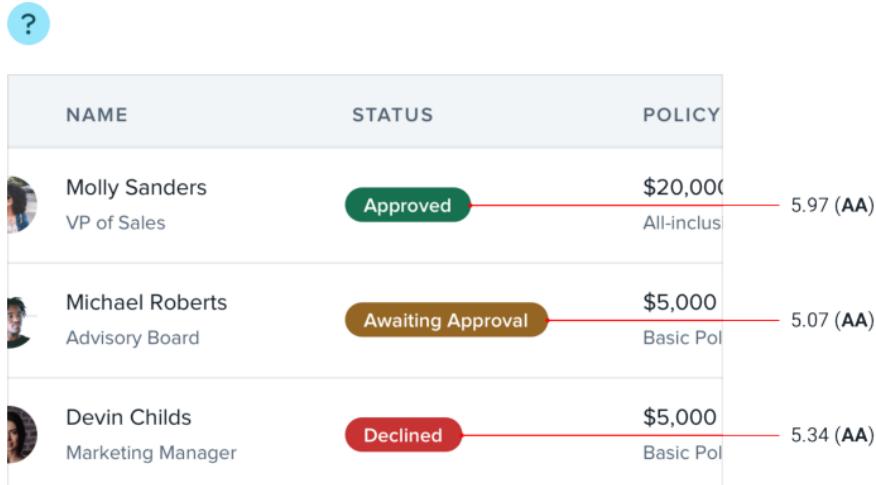
Untuk keperluan teks hitam diatas latar belakang yang cerah, memenuhi persyaratan perbandingan kontras yang direkomendasikan merupakan hal yang mudah. Namun, untuk memenuhi persyaratan yang direkomendasikan akan terasa sulit jika menggunakan warna. Ketika menggunakan teks putih diatas tampilan berwarna diperlukan warna yang lebih gelap untuk memenuhi persyaratan kontras 4.5:1.



NAME	STATUS	POLICY	
Molly Sanders VP of Sales	Approved	\$20,000 All-inclus	2.25 (Fail)
Michael Roberts Advisory Board	Awaiting Approval	\$5,000 Basic Pol	1.56 (Fail)
Devin Childs Marketing Manager	Declined	\$5,000 Basic Pol	3.14 (Fail)

Gambar 2.35: Teks putih diatas tampilan berwarna

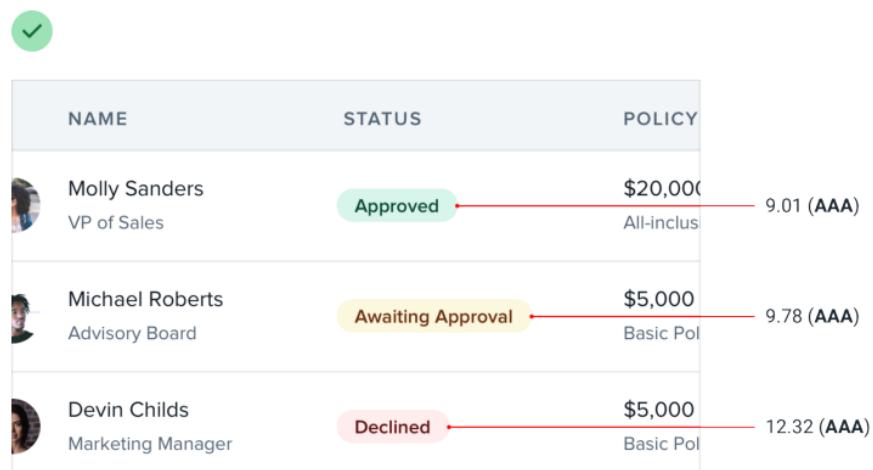
Penambahan kegelapan warna dapat menimbulkan masalah hierarki kepada elemen yang seharusnya tidak menjadi fokus utama dari halaman. Latar belakang berwarna yang gelap akan mencuri perhatian pengguna.



NAME	STATUS	POLICY
Molly Sanders VP of Sales	Approved	\$20,000 All-inclus 5.97 (AA)
Michael Roberts Advisory Board	Awaiting Approval	\$5,000 Basic Pol 5.07 (AA)
Devin Childs Marketing Manager	Declined	\$5,000 Basic Pol 5.34 (AA)

Gambar 2.36: Tampilan warna yang gelap dapat mencuri perhatian pengguna

Permasalahan ini dapat diatasi dengan membalikan kontras. Daripada menggunakan teks dengan warna terang pada latar belakang berwarna gelap, menggunakan teks dengan warna gelap diatas latar belakang berwarna cerah merupakan pilihan yang tepat.

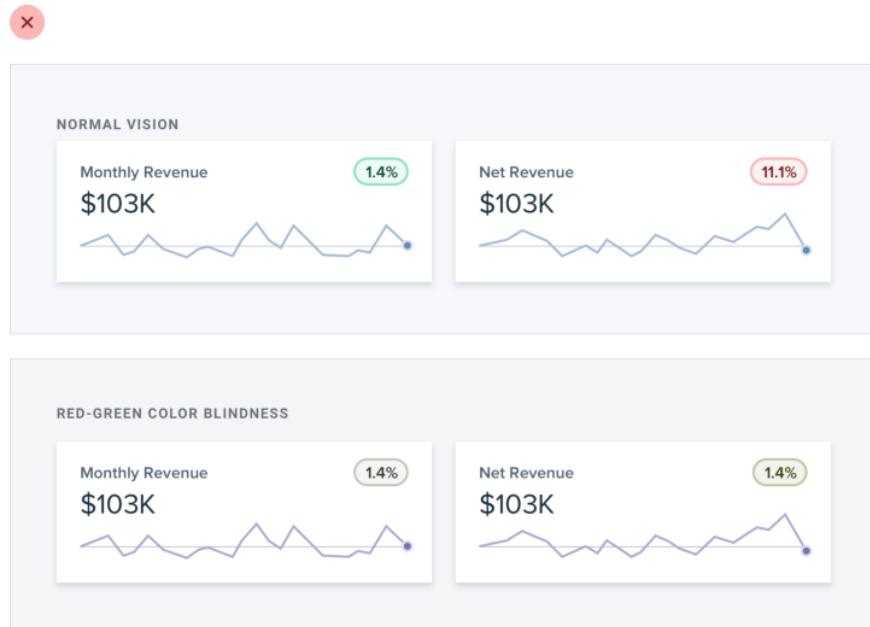


NAME	STATUS	POLICY
Molly Sanders VP of Sales	Approved	\$20,000 All-inclus 9.01 (AAA)
Michael Roberts Advisory Board	Awaiting Approval	\$5,000 Basic Pol 9.78 (AAA)
Devin Childs Marketing Manager	Declined	\$5,000 Basic Pol 12.32 (AAA)

Gambar 2.37: Pembalikan kontras antara teks dengan latar belakang

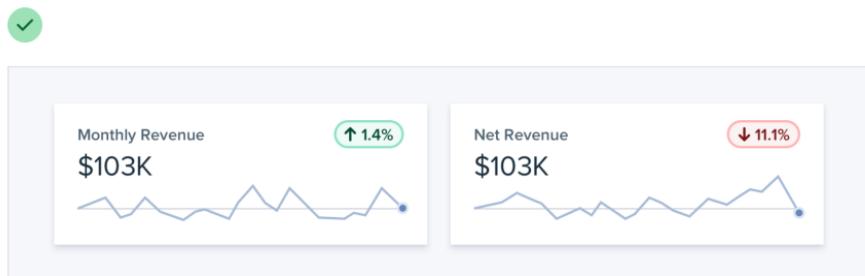
Untuk menyampaikan informasi kepada pengguna, warna saja tidaklah cukup

dalam menyampaikan pengguna atau tidak pengguna dengan penyakit buta warna akan merasa kesulitan dalam menginterpretasikan tampilan yang ada. Sebagai contoh dalam gambar statistik di bawah ini, pengguna dengan buta warna hijau akan memiliki kesulitan dalam menentukan apakah statistik tersebut membaik atau memburuk.



Gambar 2.38: Warna saja tidak cukup dalam menyampaikan informasi

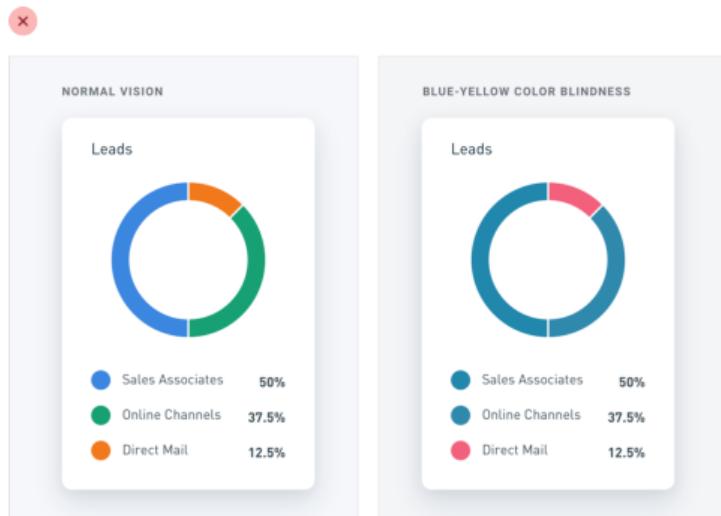
Untuk mengatasi masalah ini adalah dengan menambahkan cara lain untuk menyampaikan maksud kepada user, seperti menambahkan ikon yang mengindisikan perubahan positif atau negatif.



Gambar 2.39: Penambahan ikon disamping warna untuk menyampaikan informasi

Untuk tampilan grafik yang terkadang memiliki banyak warna yang berbeda

untuk setiap *trend line*, akan lebih baik menggunakan perbedaan kontras daripada mengandalkan warna yang berbeda untuk setiap *trend line*. Pengguna buta warna akan lebih mudah untuk mengenali perbedaan terang dan gelap dibandingkan dengan membedakan dua warna yang berbeda.



Gambar 2.40: Menggunakan warna berbeda untuk setiap *trend line*



Gambar 2.41: Menggunakan kontras yang berbeda untuk setiap *trend line*

E. Javascript

Javascript merupakan bahasa pemrograman tingkat tinggi dan bahasa pemrograman *interpreted* yang biasanya digunakan untuk menambahkan interaktivitas dan sifat dinamis dalam sebuah *websites*. Javascript biasa dikenal sebagai "Bahasa Web" dikarenakan javascript didukung oleh mayoritas *web browsers* dan dapat membuat elemen yang interaktif, memanipulasi konten sebuah *website* dan merespon aksi pengguna.

Javascript di kembangkan oleh Brendan Eich pada tahun 1995 ketika Brendan Eich sedang bekerja pada perusahaan Netscape Communications Corporation. Pengembangan javascript dilakukan karena adanya kebutuhan bahasa *scripting* yang dapat menambahkan interaksi dan fungsi dinamis dalam *website*. Pada saat itu, kebanyakan *website* merupakan statis dan kekurangan kemampuan dalam merespon interaksi pengguna.

Sebelumnya bernama "Mocha", bahasa ini kemudian berganti nama menjadi "Livescript" dan pada akhirnya "Javascript" untuk kebutuhan marketing, memanfaatkan popularitas bahasa pemrograman Java pada saat itu. Meskipun Javascript dan Java memiliki nama yang hampir mirip, keduanya merupakan bahsa yang berbeda dengan prinsip desain dan tujuan yang berbeda.

Seiring berkembangnya popularitas javascript, *web browser* lainnya seperti Microsoft dengan Internet Explorer dan Mozilla dengan Firefox, mengimplementasikan dukungan untuk bahasa pemrograman javascript. Adopsi javascript ke dalam *web browser* terkenal tersebut membuat pendukung yang kuat bagi javascript untuk menjadi standar *de facto* untuk *client-side scripting* pada *website*.

Selama bertahun tahun, Javascript telah mengalami perubahan yang signifikan dengan adanya fitur-fitur baru, improvisasi dan penambahan pada bahasa Javascript. AJAX atau *Asynchronous Javascript and XML* pada awal tahun 2000 mengembangkan kemampuan Javascript lebih jauh lagi dengan kemampuannya melakukan komunikasi *asynchronous* dengan *server* melakukan perubahan data yang dinamis pada halaman *website* tanpa melakukan muat ulang halaman *website*.

Dalam akhir akhir ini, Javascript telah berkembang melebihi dari kebutuhan aslinya. Dengan pengenalan NodeJS, program yang dapat menjalankan Javascript di luar browser. Dengan ini, Javascript dapat digunakan dalam berbagai bidang seperti *server-side scripting*, *command-line tools* dan membuat aplikasi. Javascript juga telah

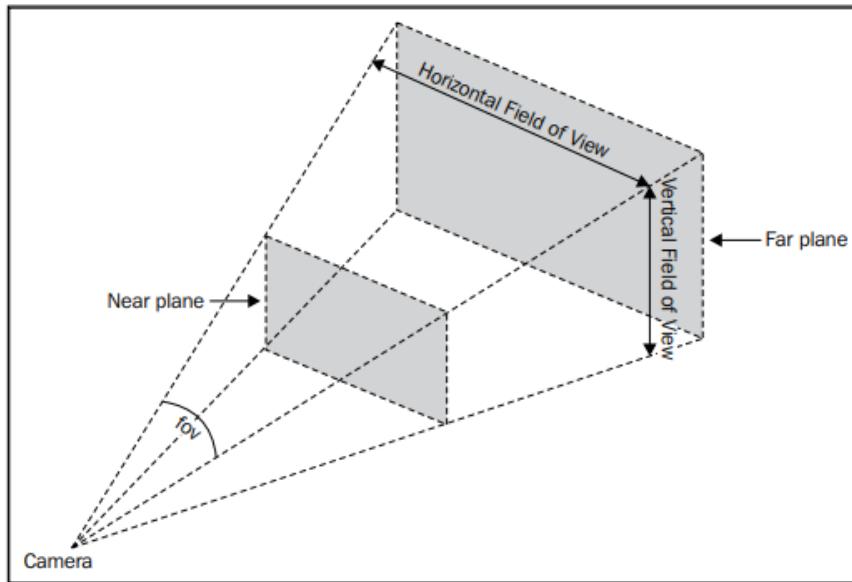
tumbuh lebih signifikan lagi dengan perkenalan berbagai *framework*, *library* dan *tools* yang mensimplifikasi proses pengembangan. *Framework* popular seperti Angular, React dan VueJS telah diadopsi oleh banyak orang, memberikan pengembang alat untuk membuat aplikasi *web* yang kompleks.

F. Three JS

Three.js merupakan sebuah library dari Javascript untuk membuat dan menampilkan grafik 3D pada web browser. Dengan menggunakan Three JS, pembuatan tampilan 3 dimensi lebih mudah untuk dilakukan dikarenakan Three JS sudah menyediakan beberapa fitur seperti *scenes*, *camera*, *lights*, *shadows*, *materials*, *textures*, *3d math* dan lain lain.

1. *Camera*

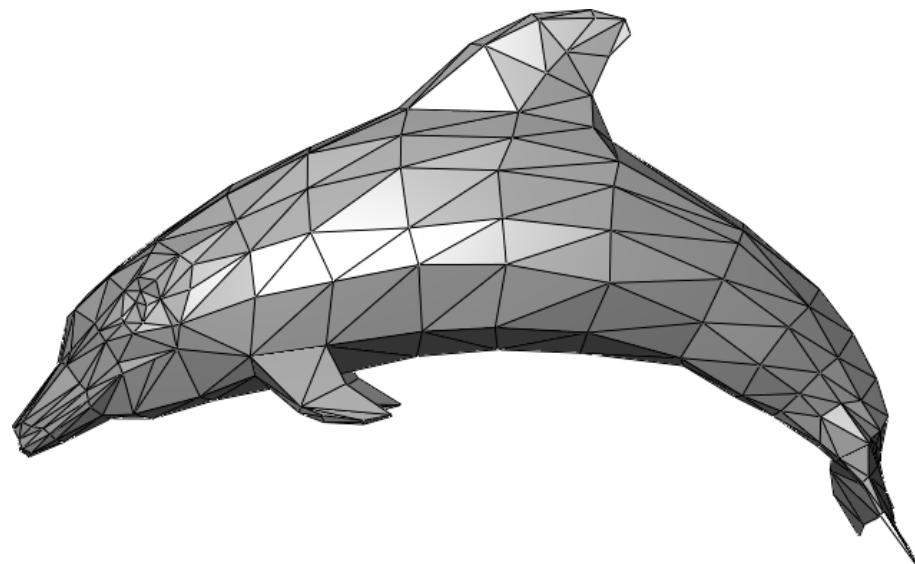
Camera mendefinisikan bagaimana apa yang akan kita lihat jika kita me-render *scene*. Jenis *camera* yang paling banyak digunakan dalam three js adalah *perspective camera* yang memberikan efek 3d dimana objek dekat terlihat lebih besar sedangkan objek yang letaknya jauh terlihat lebih kecil. *Perspective camera* mendefinisikan frustumnya melalui empat properties yaitu *near* yang mendefinisikan dimana letak frustum depan, *end* mendefinisikan dimana itu berakhir, *fov* atau *field of view* merupakan bagian dari *scene* yang dapat terlihat dari *camera* dan *aspect* merupakan perbandingan antara garis vertikal dan horizontal dari *scene* yang akan ditampilkan.



Gambar 2.42: Frustum Camera ThreeJS

2. *Mesh*

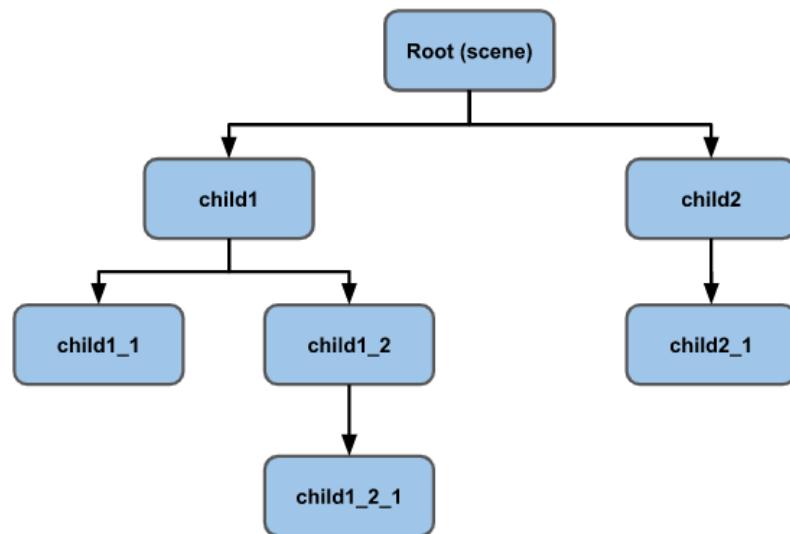
Mesh merupakan objek triangular polygon yang dibuat dengan gabungan antara objek Geometry dan Material. Mesg juga merupakan basis objek dari berbagai objek mesh lainnya seperti Skinned Mesh, MorhAnimMesh dan lain lain.



Gambar 2.43: Mesh ThreeJS

3. Scene Graph

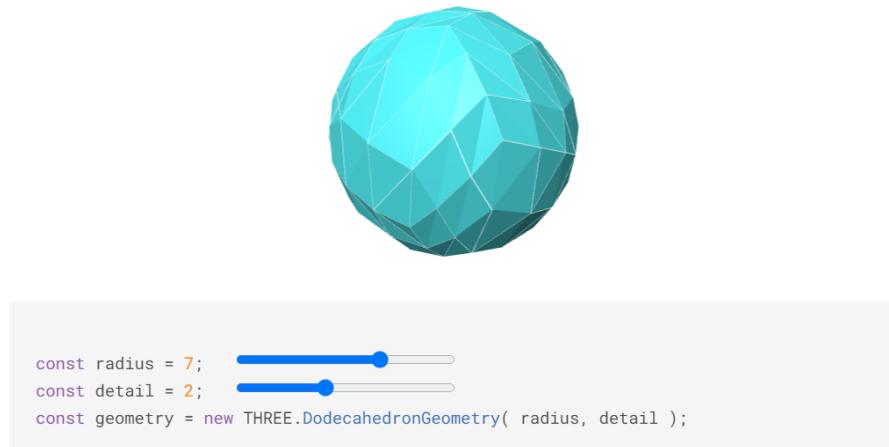
Scene graph merupakan struktur seperti pohon yang terdiri dari beberapa objek seperti *scene*, *camera*, *mesh* dan lain lain. Objek objek tersebut berstruktur secara hierarki dengan hubungan seperti *parent* dan *child* dan merepresentasikan dimana objek akan muncul dan bagaimana mereka berorientasi. *Children* diposisikan dan diorientasikan sesuai dengan *parent* mereka.



Gambar 2.44: Scene Graph ThreeJS

4. Primitives

primitives merupakan bentuk 3-dimensi yang dihasilkan saat program berjalan dengan beberapa parameter yang telah ditentukan. Three JS memiliki beberapa *primitives* bawaan seperti *BoxGeometry*, *CircleGeometry*, *CylinderGeometry*, *ConeGeometry* dan lain lain dengan parameter yang dapat ditentukan untuk setiap masing masing bentuk sesuai dengan keinginan.



Gambar 2.45: Primitives ThreeJS

5. *Drag Controls*

Pustaka ThreeJS menyediakan berbagai macam *control*, salah satunya adalah *Drag Controls*. *Drag Controls* memungkinkan pengguna menggunakan fitur *drag and drop* untuk tampilan objek tiga dimensi pada tampilan. *Drag Controls* tidak termasuk dalam fungsi yang disediakan secara langsung oleh ThreeJS melainkan fitur *Drag Controls* ini tersedia bagi user dalam sebuah addon yang harus di-*import* secara manual oleh pengguna.

G. *Unit Testing*

Pengujian perangkat lunak merupakan sesuatu yang penting untuk memastikan perangkat lunak yang dibuat dapat berjalan sesuai dengan fungsionalitas yang diharapkan. Pengujian sendiri merupakan elemen penting untuk memastikan kualitas perangkat lunak yang baik dan merupakan bagian yang tidak dapat dipisahkan dari siklus hidup pengembangan perangkat lunak seperti halnya analisis, desain, dan pengkodean (Mingtao, 2010).

Unit testing merupakan salah satu teknik yang biasa digunakan untuk melakukan pengujian perangkat lunak yang berfokus pada bagian terkecil dari sebuah aplikasi. *Unit testing* memungkinkan untuk menguji perangkat lunak secara terpisah dengan menguji bagian unit terkecil dan beberapa potongan kode seperti fungsionalitas perangkat lunak. Cara yang biasa dilakukan untuk *unit testing* adalah menulis kode *Unit testing* dan *Test case* secara manual ketika akan melakukan

pengujian. Menurut (M. Shalahuddin, 2014) “pengujian unit fokus pada usaha verifikasi pada unit yang terkecil pada desain perangkat lunak (komponen atau modul perangkat lunak). Setiap unit perangkat lunak diuji agar dapat diperiksa apakah aliran masukan (*input*) dan keluaran (*output*) dari unit sudah sesuai dengan yang diinginkan. Pengujian unit biasanya dilakukan saat kode program dibuat”

H. *User Acceptance Testing*

User Acceptance Testing merupakan pengujian yang dilakukan oleh *end-user* dimana *user* tersebut adalah staff/karyawan perusahaan yang langsung berinteraksi dengan sistem dan dilakukan verifikasi apakah fungsi yang ada telah berjalan sesuai dengan kebutuhan/fungsinya. Setelah dilakukan sistem testing, *user acceptance testing* menyatakan bahwa sistem perangkat lunak memenuhi persyaratan (Perry, 2006). Kegiatan *user acceptance testing* dapat dilakukan di ujung akhir penggerjaan suatu projek ataupun di akhir dari setiap iterasi penggerjaan yang dilakukan (Satzinger dkk., sher).

User Acceptance Test memastikan bahwa aplikasi yang akan dibuat akan memenuhi harapan dari pengguna dan bekerja dengan baik sesuai dengan apa yang diharapkan oleh pengguna. Dapat disimpulkan bahwa *user acceptance testing* adalah pengujian yang dilakukan oleh *user* atau pengguna yang dimana adalah staff/karyawan dari sebuah sistem untuk memastikan fungsi-fungsi yang ada pada sistem yang telah dibuat tersebut telah berjalan dengan baik dan sesuai dengan kebutuhan pengguna.

I. Visualisasi Data

Penggunaan kata visualisasi berasal dari latin yaitu "visualis" dan memiliki arti yaitu menggambarkan, obbservasi dan menyajikan hasil visual dari observasi dan analisis informasi digital atau suatu peristiwa. Visualisasi data merupakan proses menginterpretasi hasil analisis dengan berbagai cara untuk mewujudkan pengambilan keputusan yang lebih efisien.

Tabel merupakan metode yang sudah lama digunakan untuk mengklasifikasikan, mengorganisir dan menyajikan informasi kuantitaif dan kualitatif. Salah satu tujuan penggunaan tabel adalah untuk menampilkan data kuantitatif dengan menunjukkan hubungan simpel antara nilai kuantitatif dan kategori

yang mana nilai ini terhubung sehingga nilai nilai tersebut dapat diletakan dan dihubungan secara sendiri sendiri. Tabel memungkinkan menampilkan data dalam jumlah banyak dalam ruang yang sedikit, membuat pembaca dapat melihat keseluruhan data yang banyak dengan cepat. Beberapa konsep basik dari desain tabel adalah (Stabina, 2005)

1. Hubungan ditampilkan dalam tabel dibagi menjadi dua yaitu: *quantitative-to-categorical* digunakan untuk melihat satu data kuantitatif dalam satu waktu dan *quantitative-to-quantitative* untuk memperlihatkan hubungan antara data data.
2. Tabel dapat didesain dalam dua cara yaitu: (1) *unidirection* yaitu dimana kategori dalam bentuk baris atau kolom, tidak dapat keduanya (2) *bidirectional*, atau disebut multidirectional yang dimana lebih dari dua pasang kelompok kateogri.
3. Semua teks dalam tabel haruslah disusun secara horizontal. Judul kolom harus diulang setiap ada kelompok baru dalam kasus dimana tabel melebihi dari satu halaman. *Text alignment* dalam tabel numerikal haruslah konsisten untuk menunjukan data secara jelas.

Grafik menerjemahkan data kedalam bentuk objek visual dan merupakan alat yang kuat untuk menyampaikan informasi kuantitatif. Grafik digunakan saat dimana ketika sulit untuk mempresentasikan pola, tren atau hubungna informasi dalam bentuk verbal atau dalam bentuk tabel. Ada beberapa type grafik yang biasanya digunakan, yaitu: (Stabina, 2005)

1. Grafik Batang, menurut Bigwood dan Spore merupakan grafik dalam bentuk kolom dan batang yang disusun secara vertical maupun horizontal dan dirancang untuk mempresentasikan hubungan antara dua atau lebih pasangan data.
2. Garif garis, menurut Few mempresentasikan informasi berupa garis dan sangat cocok untuk memvisualisasikan bagaimana nilai data berubah setiap waktu, menampilkan kontinuitas, alur dan fluktuasi nilai
3. Grafik pie didesain untuk memvisualisasikan proporsi atau sebagian dari keseluruhan

Dalam pembuatan grafik ini, ada beberapa elemen desain grafik atau dapat disebut *non-data elements*, biasanya elemen desain ini dipergunakan untuk memperhidup tampilan grafik, kebutuhan artistik, sebagai dekorasi dan lain lain. jika tidak diperhatikan dengan baik, penggunaan berlebihan elemen non data ini bisa berujung menjadi "*chartjunk*". Berikut ini adalah beberapa prinsip penting dalam perencanaan dan pendesainan grafik: (Stabina, 2005)

1. Elemen grafik seperti *axis* dan *grids* bertujuan sebagai struktur pendukung dan mendefinisikan dimana data harus ditampilkan. Oleh karenanya komponen ini tidaklah harus dibuat mencolok mengalihkan perhatian dari data, elemen ini seharusnya hanya ditampilkan seminimal mungkin untuk melakukan fungsinya saja.
2. *Fills atau pattern* haruslah dipilih secara hati hati, karena hal ini jika tidak dipilih secara hati hati dapat mengakibatkan distraksi atau misinterpretasi data yang disajikan. Penggunaan elemen seperti (*stripes*, *weaves*, *checkers*, *dots*) membuat ilusi yang dapat disebut *fabric effect*
3. Perhatian khusus harus diberikan kepada pemberian efek tiga dimensi untuk mempresentasikan data yang dimana penggunaan efek ini menjadi luas karena fitur ini disediakan oleh perangkat lunak *spreadsheet* konvensional yang beredar di pasaran. Kebanyakan peneliti setuju bahwa penggunaan efek ini haruslah dihindari.
4. Menurut Bigwood dan Spore, Pelabelan data yang sesuai juga memainkan peran penting dalam mempresentasikan data secara grafikal dan aspek seperti jarak antara elemen grafik, orientasi horizontal teks, dan penggunaan kalimat yang jelas juga penting dalam menyajikan informasi yang akurat. Penggunaan legenda jugalah harus diperhatikan. Beberapa penulis berpendapat bahwa penggunaan legenda sebaiknya digunakan ketika kasus dimana label data terlalu panjang untuk dimuat dalam elemen grafik. Beberapa penulis juga setuju peletakan komponen legenda ini haruslah sedekat mungkin dengan grafik.

J. Participatory Design

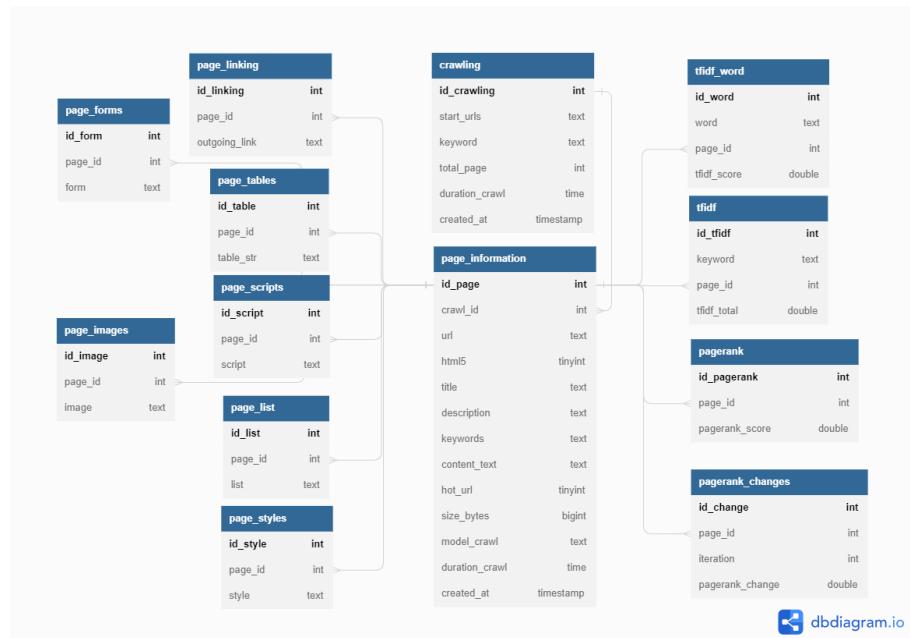
Participatory design merupakan suatu kumpulan teori, praktis dan ajaran yang mengikuti sertakan dengan pengguna dalam aktivitas yang mengarah ke produk

perangkat lunak dan perangkat keras (Muller, 2007). Banyak peneliti dan praktisi dalam *Participatory design* termotivasi dalam improvisasi proses internal dan kombinasi dari berbagai pengetahuan untuk membuat pelayanan dan produk yang lebih baik.

Participatory design bermula dari Scandinavia pada tahun 1970-an dan 1980-an. Karya Scandinavian ini dipicu oleh komitmen dari Marxist untuk memberdayakan pekerja dan membina demokrasi dalam tempat kerja. Persatuan buruh mempunyai pengalaman yang rendah mengenai teknologi komputer dan terpaksa diharuskan untuk menerima sistem yang dikembangkan oleh manajemen, sistem yang merepresentasikan ketidaksesuaian dengan cara kerja pekerja pada biasanya. Oleh karena mereka tidak mengerti dalam mendesain sebuah teknologi komputer mereka sendiri, para pekerja ditempatkan dalam posisi menerima atau menolak teknologi yang ada. Beberapa peneliti Scandinavia mengutarakan cara pengembangan, sebuah pendekatan yang menyajikan "*language games*"

K. Analisa Desain Database

Teknologi database yang digunakan untuk penelitian search engine (Khatulistiwa, 2022) adalah MySQL yang merupakan database SQL. Adapun relasi tabel tabel yang digunakan pada penelitian sebelumnya adalah sebagai berikut.



Gambar 2.46: Skema database yang digunakan dalam proses crawling

Pada saat proses *crawling* berjalan, *crawler* membutuhkan beberapa tabel untuk menyimpan informasi yang didapatkan *crawler* dalam proses berjalannya. Tabel-tabel tersebut diantaranya adalah: *page linking* merupakan tabel untuk menyimpan link keluar dari suatu halaman, *page images* digunakan untuk menyimpan semua gambar yang ada dalam suatu halaman, *page scripts* digunakan untuk menyimpan semua *script* yang terdapat dalam suatu halaman, *page tables* digunakan untuk menyimpan semua tabel yang terdapat pada suatu halaman, *page styles* digunakan untuk menyimpan *style* yang bertugas untuk memberi tampilan pada suatu halaman, *page forms* untuk menyimpan semua form yang terdapat dalam suatu halaman dan *page list* untuk menyimpan seluruh *list* dalam suatu halaman. Masing masing dari tabel yang telah disebutkan terbentuk berhubungan *Many to one* dengan tabel *page information*. Sedangkan untuk tabel yang digunakan untuk *document ranking* dan *page ranking* adalah tabel *tfidf*, tabel *tfidf word* dan tabel *pagerank*. Untuk menyimpan informasi awal guna untuk memulai proses crawling digunakan tabel *crawling* sebagai tempat penyimpanannya. Adapun konsumsi penyimpanan yang dikonsumsi oleh masing masing tabel ditampilkan sebagai berikut.

table name	storage usage	rows
crawling	16K	7
page_forms	6.7M	11733
page_information	34M	10736
page_linking	144M	1533925
page_list	278M	941820
page_scripts	462M	464502
page_styles	16M	34565
page_tables	14M	17528
pagerank	624K	10714
tfidf	1.8M	13774
tfidf_word	95M	1682007
page_images	106M	632677

Gambar 2.47: Penggunaan storage masing masing tabel dalam database

L. Celery

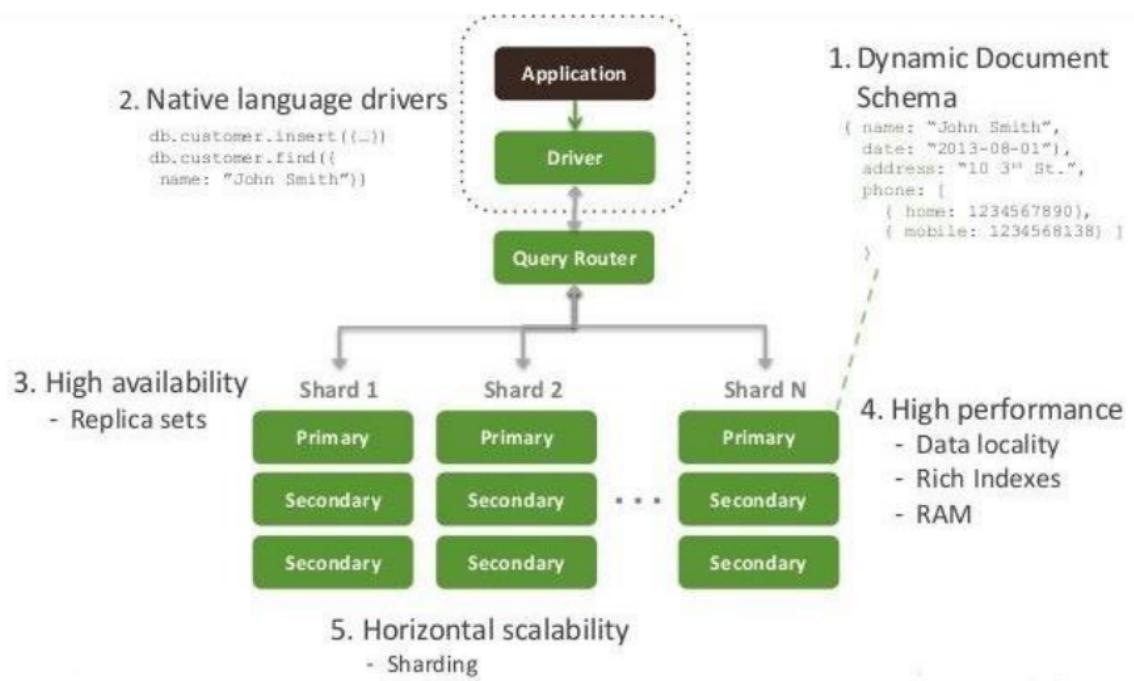
Celery merupakan *framework task queue* yang ditulis dengan menggunakan bahasa pemrograman *python*. Program *Celery* dapat membantu menangani tugas tugas yang berkaitan dengan mengatur pembagian kerja dalam beberapa pekerja.

M. Mongo DB

MongoDB dikembangkan pada tahun 2007 oleh Eliot dan Dwight untuk 10gen, yang merupakan sistem basis data yang berbasis dokumen yang dapat berjalan di berbagai platform. MongoDB termasuk basis data yang jatuh pada kategori NoSQL. Salah satu alasan dibuatnya MongoDB adalah kemampuan manajemen data dalam jumlah besarnya (Mungekar, 2019). MongoDB menyimpan dokumen dalam bentuk seperti JSON atau *Javascript Object Annotation* yang bentuknya dapat bermacam macam. Informasi yang relevan dapat disimpan secara bersama sama untuk *query* dengan akses yang cepat dengan *MongoDB query language*. MongoDB menggunakan skema yang dinamis yang membantu dalam membuat *record* tanpa mendefinisikan strukturnya terlebih dahulu seperti atribut atau tipe data. MongoDB memungkinkan untuk mengubah struktur data dengan mudah dengan cara menambahkan atribut atau menghapus atribut yang ada. Model

penyimpanan seperti ini membuat membantu dalam merepresentasikan hubungan hierarki, menyimpan data *arrays* dan struktur yang kompleks lainnya dengan mudah. Sebuah dokumen dalam suatu *record* tidak diharuskan memiliki atribut yang sama. MongoDB dirancang untuk availabilitas tinggi dan skalabilitas yang termasuk *replication* dan *auto-sharding* (B dkk., 2016).

MongoDB mendukung dua tipe replikasi yaitu *master-slave* dan *replica sets*. Dalam replikasi *master-slave*, *master* mempunyai akses data penuh dan menentukan siapa yang berhak menulis setiap perubahan kepada *slave*. Para *slave* dalam tipe replika ini hanya dapat membaca data saja. Untuk *replica sets*, memiliki cara kerja yang sama dengan replikasi *master-slave*, akan tetapi *replica sets* memungkinkan untuk memilih *master* yang baru jika *master* yang lama tidak dapat melakukan kewajibannya. Fitur lainnya yang didukung oleh MongoDB adalah *automatic sharding*. Dengan menggunakan fitur ini, data dapat dibagi-bagi ke beberapa *node*. Seseorang harus memverifikasi *sharding key* untuk setiap *collection* yang mendefinisikan bagaimana suatu dokumen dapat dibagi-bagi. (B dkk., 2016)



Gambar 2.48: Arsitektur MongoDB

Pada umumnya dalam arsitektur MongoDB terdapat beberapa komponen pendukung seperti:

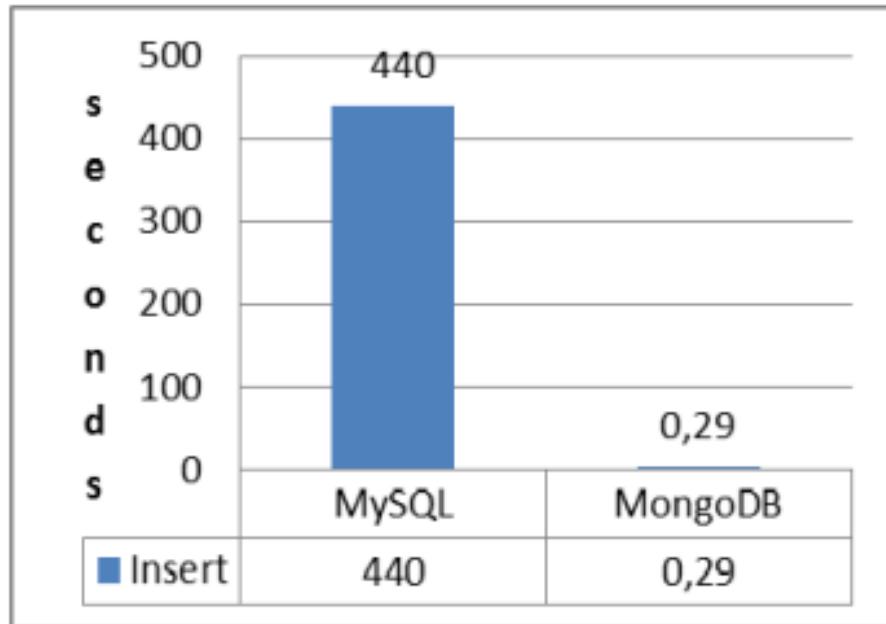
1. *Configuration Servers* bertugas untuk menyimpan metadata dari *sharded cluster*. Data yang disimpan mengandung informasi berupa *mapping dataset cluster* dengan *shard*. Komponen seperti *query router* menggunakan data ini untuk menargetkan operasi yang datang ke shard yang sesuai.
2. *Query Routers* berhadapan langsung dengan aplikasi dan mengalihkan operasi ke shard atau shards yang sesuai.
3. *Shards* menyimpan data dan memberikan availabilitas dan konsistensi data yang tinggi.

Dibandingkan dengan MySQL, MongoDB lebih baik dalam hal pemrosesan query (B dkk., 2016). Hal ini dibuktikan dengan penelitian yang bertujuan untuk melihat kelebihan antara penggunaan database non-relasional MongoDB dengan database relasional MySQL. Pada penelitian ini dilakukan beberapa operasi basik kepada kedua database yaitu MongoDB dan MySQL. Operasi yang dilakukan yaitu *insert*, *select*, *update* dan *delete*. Penelitian dilakukan dengan perangkat keras seperti Windows 7 Ultimate 64-bit, prosessor Intel Core i3 (2.4 GHz), 4 GB RAM memory. Sebelum penelitian dilakukan kedua database memiliki beberapa tabel denngan skema yang sama yaitu:

1. tabel/dokumen *User* dengan kolom id, username, password, email.
2. tabel/dokumen *Forum* dengan kolom: id, title, author, info (short description).
3. tabel/dokumen *Subforum* dengan kolom: id, title, author, info, created, updated.
4. tabel/dokumen *Discussion* dengan kolom: id, title, author, created, updated, content.
5. tabel/dokumen *Comments* dengan kolom: id, author, created, content, approved.

Pemasukan data dimulai dengan memasukan data pengguna ke kedua database, 10.000 pengguna dimasukan ke dalam dua database. Untuk id user dihasilkan secara automatis oleh kedua database dan untuk atribut seberti *username*, *password* dan alamat *email* digunakan fungsi PHP seperti *md5*, *rand*, *substr* dan *str shuffle*. Setelah pemasukan data dilakukan, MySQL memerlukan waktu selama 440 detik sedangkan MongoDB memerlukan waktu sekitar 0.29 detik. Untuk pengujian

performa *query* data dalam jumlah besar, terlihat MongoDB lebih performan daripada MySQL.



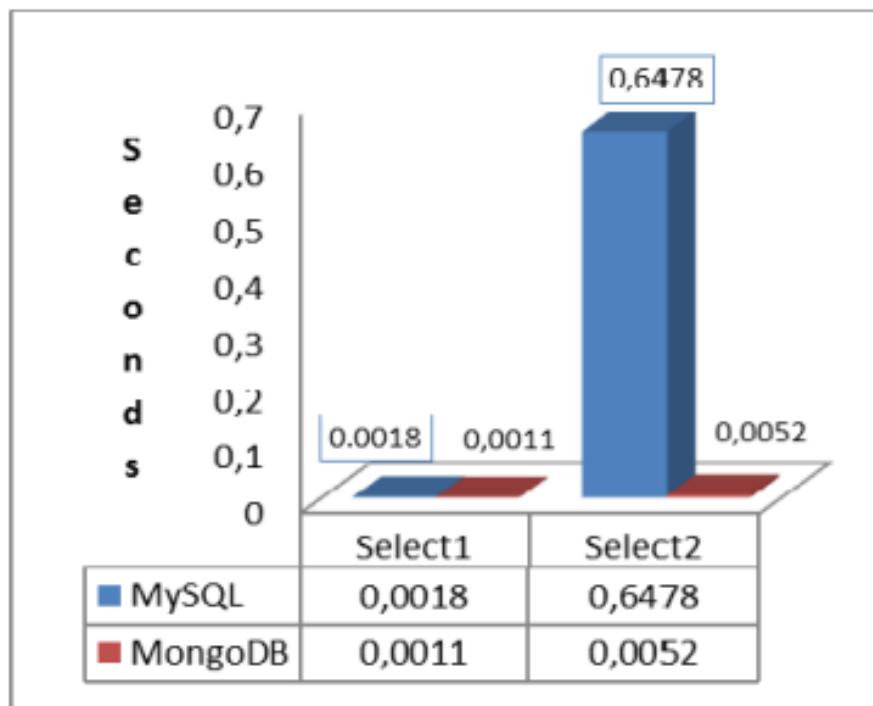
Gambar 2.49: Perbandingan performa pemasukan data pengguna untuk kedua database

Ketika semua pengguna berhasil dimasukan kedalam database, pemasukan data dilanjutkan dengan pemasukan data forum, subforum, diskusi dan komentar. Untuk pengujian, dilakukan pemasukan 5000 baris data untuk setiap tabel forum, subforum, diskusi, dan komentar. Setelah pemasukan data dilakukan, MySQL memerlukan waktu selama 1010 detik sedangkan MongoDB memerlukan waktu sekitar 3.3331 detik. Ketika semua pengguna berhasil dimasukan kedalam database, pemasukan data dilanjutkan dengan pemasukan data forum, subforum, diskusi dan komentar. Untuk pengujian, dilakukan pemasukan 5000 baris data untuk setiap tabel forum, subforum, diskusi, dan komentar. Setelah pemasukan data dilakukan, MySQL memerlukan waktu selama 1010 detik sedangkan MongoDB memerlukan waktu sekitar 3.3331 detik. Untuk pengujian performa untuk pemasukan data dalam jumlah besar, terlihat MongoDB lebih performan daripada MySQL dalam hal memasukan data yang besar.

Selanjutnya adalah pengujian operasi *query*, diadakan dua operasi *query* yaitu:

1. *Query* pertama untuk semua diskusi yang suatu user lakukan dan dengan tanggal yang berbeda dari yang ditentukan.

2. Query kedua untuk semua pengguna dari database dan jumlah diskusi yang dimulai oleh setiap user

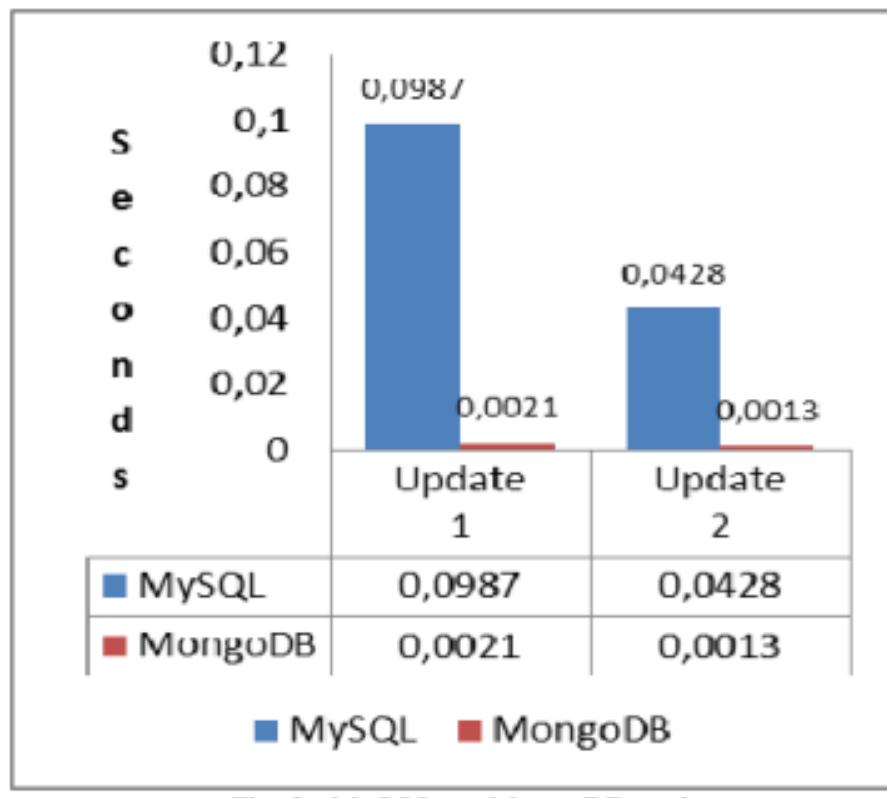


Gambar 2.50: MySQL vs MongoDB Select

Dari tabel diatas, untuk operasi query yang pertama, MySQL memerlukan waktu sekitar 0.0018 detik sedangkan MongoDB berhasil dieksekusi dalam waktu 0.0011 detik dan untuk operasi yang kedua, MySQL memerlukan waktu sekitar 0.6478 detik dan MongoDB memerlukan waktu 0.0052 detik. Untuk pengujian performa untuk query data dalam jumlah besar, terlihat MongoDB lebih performan daripada MySQL dalam hal melakukan operasi *query*.

Selanjutnya adalah pengujian operasi *update* database, ada dua operasi *update* yang digunakan dalam pengujian ini:

1. Mengupdate suatu komen yang ditulis oleh suatu user
2. Mengupdate alamat email suatu pengguna

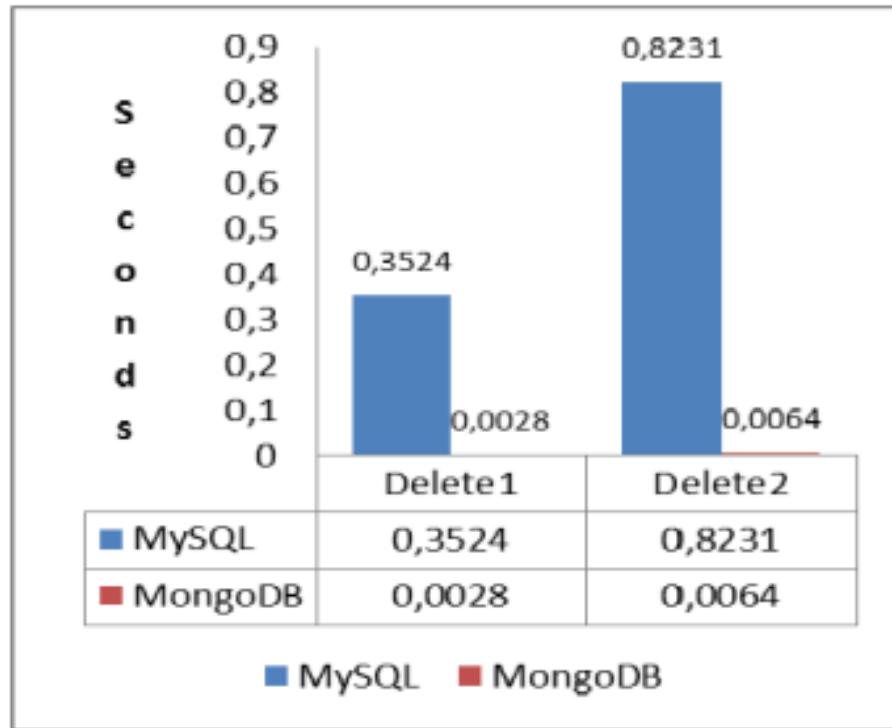


Gambar 2.51: MySQL vs MongoDB update

Untuk operasi update yang pertama, MySQL memerlukan waktu sekitar 0.0987 detik sedangkan dalam MongoDB operasi query update yang pertama dilakukan dalam 0.0428 detik. Untuk pengujian performa untuk update data dalam jumlah besar, terlihat MongoDB lebih performan daripada MySQL dalam hal melakukan operasi *update*.

Selanjutnya dilakukan pengujian untuk operasi *delete*. Ada dua operasi *delete* yang dilakukan untuk pengujian yaitu:

1. Menghapus semua forum yang dibuat oleh suatu user
2. Menghapus semua komentar yang dibuat oleh suatu user.



Gambar 2.52: MySQL vs MongoDB delete

Dari gambar diatas, MySQL memerlukan waktu sekitar 0.3524 detik, sedangkan MongoDB memerlukan 0.0028 detik dan operasi yang kedua, MySQL memerlukan waktu 0.8231 detik dan MongoDB memerlukan waktu sekitar 0.0064 detik.

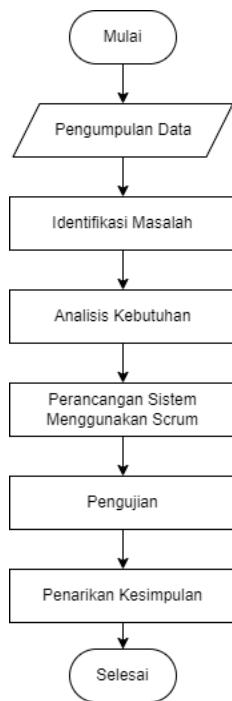
MongoDB memberikan waktu eksekusi operasi yang lebih rendah dibandingkan dengan MySQL dalam 4 operasi biasa yang dimana hal ini adalah hal yang penting ketika sebuah aplikasi diharuskan mendukung ribuan pengguna secara bersama-sama. Maka dari itu dari penelitian yang dilakukan sebelumnya, MongoDB mempunyai performa yang baik dan lebih dipilih dibandingkan MySQL (Győrödi dkk., 2015).

BAB III

METODOLOGI PENELITIAN

A. Deskripsi Penelitian

Pada penelitian ini akan merancang tampilan dan aplikasi *web* dengan *admin console* untuk *search engine* yang telah dibuat pada penelitian sebelumnya. Berikut merupakan tahapan-tahapan penelitian yang penulis akan lakukan dalam perancangan aplikasi:



Gambar 3.1: Tahapan penelitian

B. Pengumpulan Data

Peneliti melakukan studi literatur dengan membaca jurnal-jurnal, penelitian terdahulu dan buku yang berkaitan dengan topik penelitian.

C. Analisa Kebutuhan

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

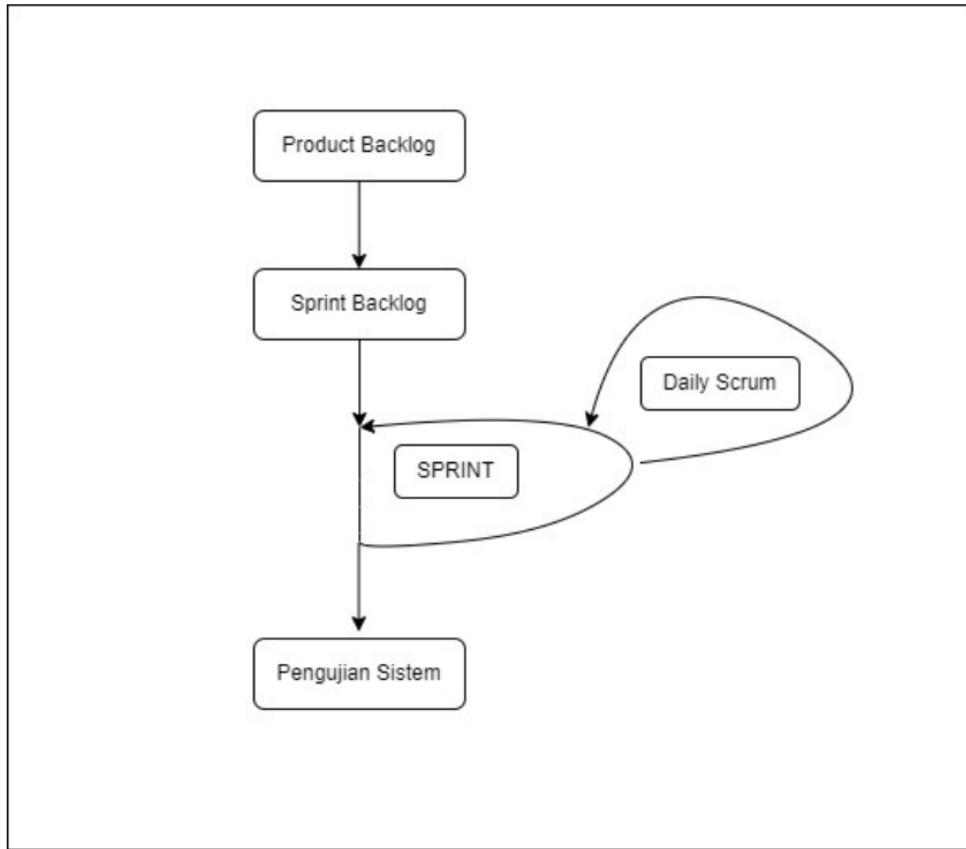
1. Laptop yang mempunyai CPU Intel Core i5-8250U 1.60 GHz (8 Cores) dengan RAM sebesar 12 GB
2. Koneksi berbasis Wi-Fi

Perangkat keras yang telah disebutkan sudah memenuhi persyaratan dalam menjalankan perangkat lunak yang akan digunakan seperti:

1. Windows 10 64 bit
2. Visual Studio Code
3. Figma
4. Adobe Illustrator CC6

D. Perancangan Sistem Dengan Scrum

Dalam penelitian ini akan digunakan metode *scrum* agar penelitian ini menjadi lebih terstruktur dan mudah. Desain penelitian akan menjelaskan tahapan tahapan yang akan dilakukan penulis dalam penelitian ini dengan menggunakan metode *scrum*.



Gambar 3.2: Desain Penelitian

Tahap pertama yang akan dilakukan pada penelitian ini adalah membuat *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.

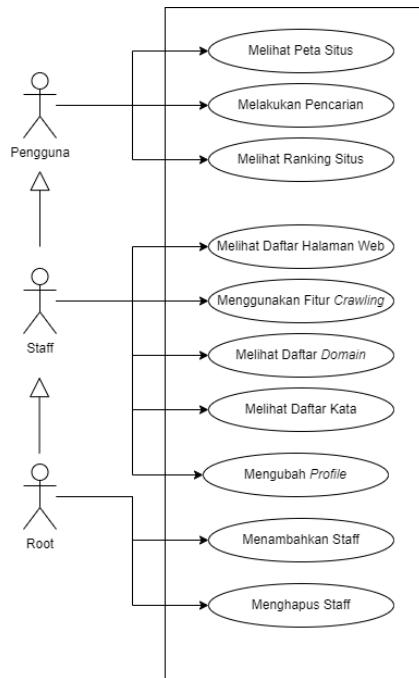
Pada penelitian ini akan menggunakan metode *scrum* sebagai metode pengembangan sistemnya. Komponen-komponen *scrum* terdiri dari *product backlog*, *sprint backlog*, *daily scrum*, *sprint* dan dilanjutkan dengan pengujian sistem yang sudah dibuat. Tahapan pertama yang dilakukan pada penelitian ini adalah mendefinisikan sebuah *product backlog* penelitian dan mendefinisikan *sprint backlog* yang merupakan jadwal pengerjaan tugas-tugas yang akan dibuat. Setelah kedua komponen tersebut berhasil dibuat, maka *sprint* dapat dimulai dari awal dengan diselingi *daily scrum* untuk melakukan pelaporan *progress* penelitian yang sedang dilakukan. Berikut ini adalah tabel *product backlog* yang telah dibuat.

Tabel 3.1: *Product Backlog*

No.	Story	Sprint	Status
1	Fitur pencarian pengguna	1, 2, 8	Selesai
2	Fitur <i>page ranking</i>	4, 5	Selesai
3	Fitur <i>staff</i>	5, 8	Selesai
4	Fitur <i>document ranking</i>	5, 7	Selesai
5	Fitur <i>crawling</i>	3, 6	Selesai
6	Struktur projek	2	Selesai
7	<i>multi-threaded service</i>	10	Selesai
8	<i>Service deployment</i>	11	Selesai
9	Background task dengan <i>Celery</i>	12	Selesai
10	Pengujian penggunaan memori	13	Selesai

Berdasarkan tabel di atas, *Product Backlog* penelitian ini terdiri dari 3 komponen yaitu *story*, *sprint* dan *status*. *Story* ialah sebuah pekerjaan besar yang dapat dibagi bagi lebih kecil lagi menjadi tugas tugas kecil. *Sprint* menandakan *sprint* berapa *story* tersebut akan diselesaikan. *Status* memberitahu apakah *sprint* tersebut sudah terlaksanakan atau belum.

E. Use Case Diagram



Gambar 3.3: Use Case Diagram

F. Pengujian Sistem

Pengujian pada *search engine* menggunakan *User Acceptance Test* (UAT). UAT akan dilaksanakan oleh pengguna dengan *scrum master* untuk mengetahui apakah aplikasi sudah sesuai kebutuhan dan layak digunakan dan *Unit Testing*.

1. Unit Testing

Unit Testing yang akan dibuat mengacu pada fitur-fitur yang akan dibuat dalam *product backlog*. Pengujian *Unit Testing* dilakukan di setiap akhir sprint yang dilakukan oleh peneliti sendiri.

2. User Acceptance Test

User Acceptance Testing dibuat berdasarkan fitur-fitur yang dapat diakses oleh pengguna pada *product backlog* yang telah dibuat sebelumnya. Untuk format *User Acceptance Test* digunakan format sebagai berikut. Kolom kesesuaian dibagi menjadi dua yaitu Setuju dan Tidak Setuju

Tabel 3.2: Format *User Acceptance Test*

No	<i>Acceptance Requirements</i>	Kesesuaian		Keterangan
		Setuju	Tidak Setuju	
1	Fitur pencarian pengguna sudah sesuai dengan kebutuhan pengguna			
2	Fitur staff sudah sesuai dengan kebutuhan pengguna			
3	Fitur <i>crawling</i> sudah sesuai kebutuhan pengguna			
4	Fitur <i>document ranking</i> sudah sesuai kebutuhan pengguna			
5	Fitur <i>page ranking</i> sudah sesuai kebutuhan pengguna			

BAB IV

HASIL DAN PEMBAHASAN

A. Sprint 1

Tabel 4.1: Sprint 1 Backlog

No	Story	Task	Status
1	Fitur pencarian pengguna	Menerapkan <i>mock-up</i> tampilan halaman hasil pencarian dan pencarian <i>search engine</i>	belum
		Membuat <i>mock-up</i> tampilan halaman hasil pencarian dan pencarian <i>search engine</i>	selesai
		Membuat <i>web service</i> untuk tampilan halaman hasil pencarian dan pencarian <i>search engine</i>	belum
2	Fitur <i>staff</i>	Menerapkan tampilan login, daftar, tambah <i>staff</i>	belum
		Membuat <i>mock-up</i> tampilan login, daftar, tambah <i>staff</i>	selesai
		Membuat <i>web service</i> untuk tampilan login, daftar, tambah <i>staff search engine</i>	belum
3	Fitur <i>page ranking</i>	Menerapkan <i>mock-up</i> tampilan status <i>page ranking search engine</i>	belum
		Membuat <i>mock-up</i> tampilan status <i>page ranking search engine</i>	belum
		Membuat <i>web service</i> untuk tampilan halaman <i>page ranking search engine</i>	belum
		Membuat <i>mock-up</i> tampilan peta situs <i>search engine</i>	selesai

		Menerapkan <i>mockup</i> tampilan peta situs	belum
--	--	--	-------

a). Penentuan Aturan Desain

Pada *sprint* ini akan dilakukan perancangan tampilan untuk *search engine ONE (Omniscience Network Extractor)* dilakukan dengan perangkat lunak Figma dan Adobe Illustrator. Untuk memudahkan proses dalam perancangan tampilan untuk *search engine* yang akan dibangun, diperlukan adanya suatu sistem desain untuk *sizing*, *spacing* dan *color* atau warna.

Dalam sistem desain untuk *spacing* dan *sizing* digunakan seperti tabel berikut

Tabel 4.2: Sistem desain sizing dan spacing

Name	Size (16px base)	Pixels	
1	0.25rem	4px	4px
2	0.5rem	8px	8px
3	0.75rem	12px	12px
4	1rem	16px	16px
5	1.25rem	20px	20px
6	1.5rem	24px	24px
8	2rem	32px	32px
10	2.5rem	40px	40px
12	3rem	48px	48px
16	4rem	64px	64px
20	5rem	80px	80px
24	6rem	96px	96px
32	8rem	128px	128px
40	10rem	160px	160px
48	12rem	192px	192px
56	14rem	224px	224px
64	16rem	256px	256px

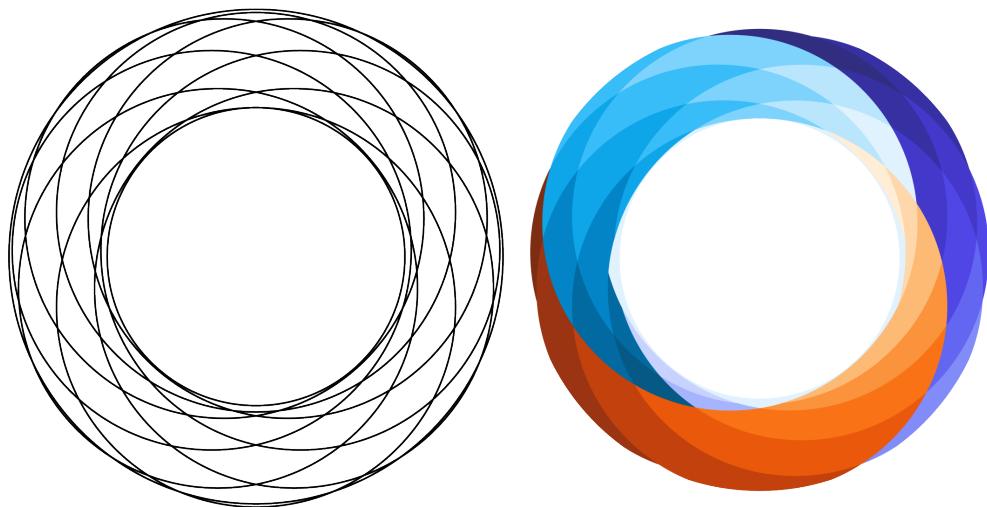
Untuk sistem desain warna, warna biru *navy* digunakan sebagai warna utama tampilan *search engine*. Adapun *shades* dari warna utama yaitu *navy* adalah

sebagai berikut. Warna *shades* ditentukan dengan cara mengatur nilai *hue*, *saturation* dan *lightning* dari warna utama.

AA 5.68 25 #F5FAFF	AA 5.55 50 #EFF8FF	AA 4.78 100 #D1E9FF	4.17 200 #B2DDFF	1.77 300 #84CAFF	2.32 400 #53B1FD	3.24 500 #2E90FA	AA 4.57 600 #1570EF	AA 5.97 700 #175CD3	AAA 800 #1849A9	AAA 900 #194185
--------------------------	--------------------------	---------------------------	------------------------	------------------------	------------------------	------------------------	---------------------------	---------------------------	-----------------------	-----------------------

Gambar 4.1: Desain Penelitian

Hal pertama yang dilakukan dalam pembuatan tampilan dari *search engine* adalah mendesain sebuah logo. Sebuah logo digunakan sebagai identitas bagi *search engine* yang akan dibuat tentulah harus mempunyai makna. Logo yang akan dibuat berbentuk "O" melambangkan huruf pertama dari nama *search engine* tersebut yaitu "ONE". Tanpa warna, logo pola jaring akan terlihat dalam lingkaran berbentuk O yang melambangkan suatu hubungan dari banyaknya sebuah situs. Untuk pewarnaan, digunakan warna oranye yang melambangkan huruf O dari nama *search engine* tersebut, warna navy melambangkan huruf N dari nama *search engine* dan warna *eclipse* melambangkan huruf terakhir dari *search engine* tersebut yaitu E.



Gambar 4.2: Desain Logo ONE *Search Engine*



Gambar 4.3: Desain sistem warna pada logo ONE

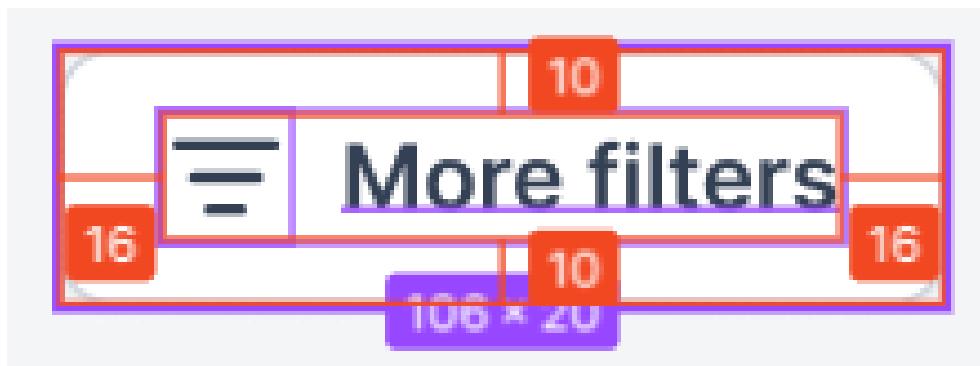
Terdapat dua bagian dari tampilan *search engine*, yaitu bagian untuk admin untuk manajemen *search engine* dan tampilan untuk *pengguna*.

Menurut (Adam dan Schoger, 2018), jenis font dapat ditentukan dengan observasi jenis *font* yang digunakan dari aplikasi lain yang sudah ada di pasaran. Pada dasarnya, aplikasi yang sudah ada di pasaran memiliki beberapa orang perancang yang memiliki pengalaman lebih mengenai *typography* untuk menentukan jenis *font* yang akan mereka gunakan biasanya dengan telah memperhitungkan berbagai faktor. Setelah melakukan observasi pada aplikasi besar yang ada pada pasaran, peneliti memutuskan untuk menggunakan *font Inter* sebagai jenis *font* utama pada aplikasi dikarenakan jenis *font* ini sudah banyak sekali digunakan pada aplikasi besar yang terdapat di pasaran dan bersifat *open source*.

Dalam penentuan *white spacing* dari setiap komponen yang akan dibuat, penulis menggunakan metode *decremental* seperti yang telah dibahas sebelumnya. Pemberian *white spacing* secara *decremental* adalah sebuah cara untuk menemukan nilai *white spacing* yang cocok digunakan pada komponen yang akan dibuat dengan cara memberikan nilai awal *white spacing* yang besar kepada komponen yang akan dibuat dan mengurangi nilai *white spacing*-nya secara bertahap sampai dapat dikatakan cocok. Dalam pemberian *white spacing* penulis menggunakan sistem desain *white spacing* dan *sizing* yang telah ditentukan sebelumnya.

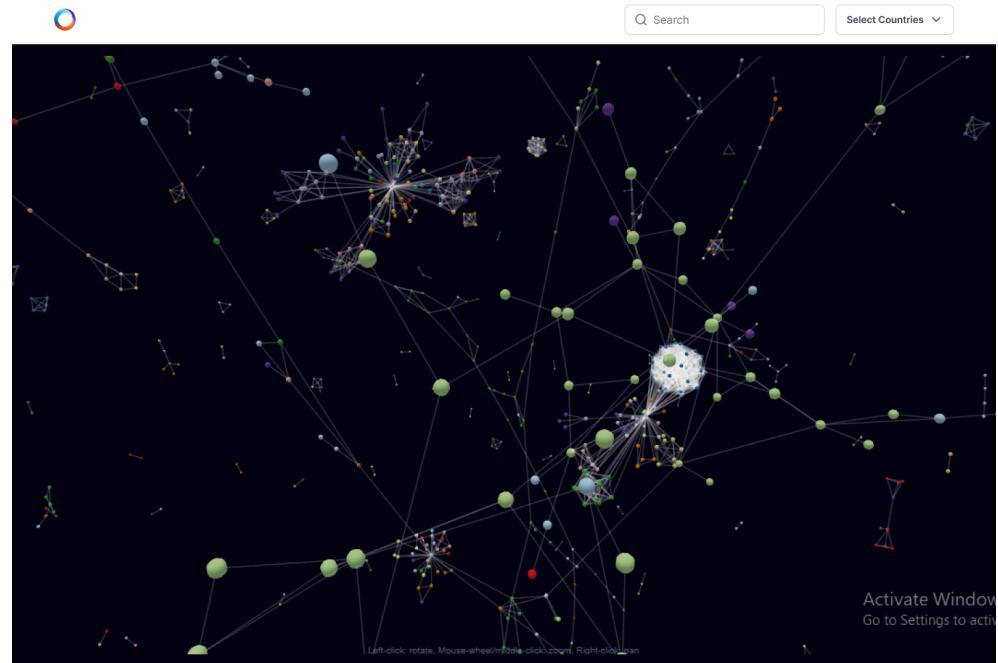


Gambar 4.4: Pemberian nilai *white spacing* awal yang besar terhadap komponen *button*



Gambar 4.5: Pemberian nilai *white spacing* secara *decremental* terhadap komponen *button* sehingga *white spacing* yang diberikan terasa cocok

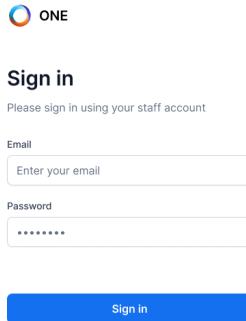
b). Merancang tampilan peta situs



Gambar 4.6: Rancangan tampilan peta situs

Pada halaman ini disajikan daftar situs dengan bentuk graf tiga dimensi.

- c). Merancang tampilan *staff*



Gambar 4.7: Rancangan tampilan peta login untuk *staff*

Pada halaman ini *staff* dapat melakukan *login* dengan akun mereka untuk masuk ke *dashboard*.

Gambar 4.8: Rancangan tampilan untuk menambahkan *staff*

Pada halaman ini *staff* dengan akses *root* dapat menambahkan staff baru

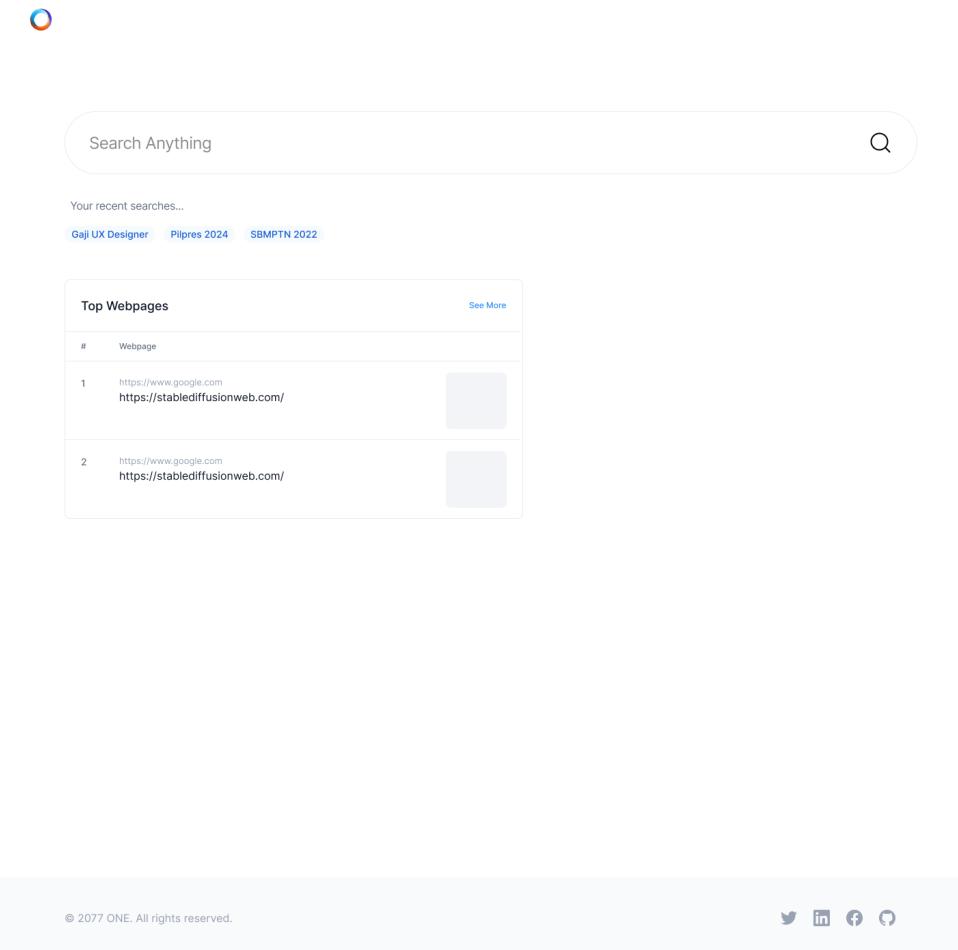
dengan mengisi formulir yang telah disediakan.

The screenshot shows the 'ONE Dashboard' interface. On the left, there's a sidebar with a 'Menu' section containing 'Overview' (which is highlighted), 'Staff', 'Documents', 'Services' (with 'Crawling Service', 'Document Ranking Service', and 'Page Ranking Service' listed under it), and a user profile icon at the top right. The main content area is titled 'Update Profile' and has a sub-section 'Page Ranking Service'. It contains a form for 'Personal Info' with fields for 'Email' (placeholder: 'Enter your email'), 'Password' (placeholder: 'Enter your password'), 'First Name' (placeholder: 'Enter your first name'), and 'Last Name' (placeholder: 'Enter your last name'). A blue 'Update Profile' button is located at the bottom right of the form.

Gambar 4.9: Rancangan tampilan mengubah profile bagi *staff*

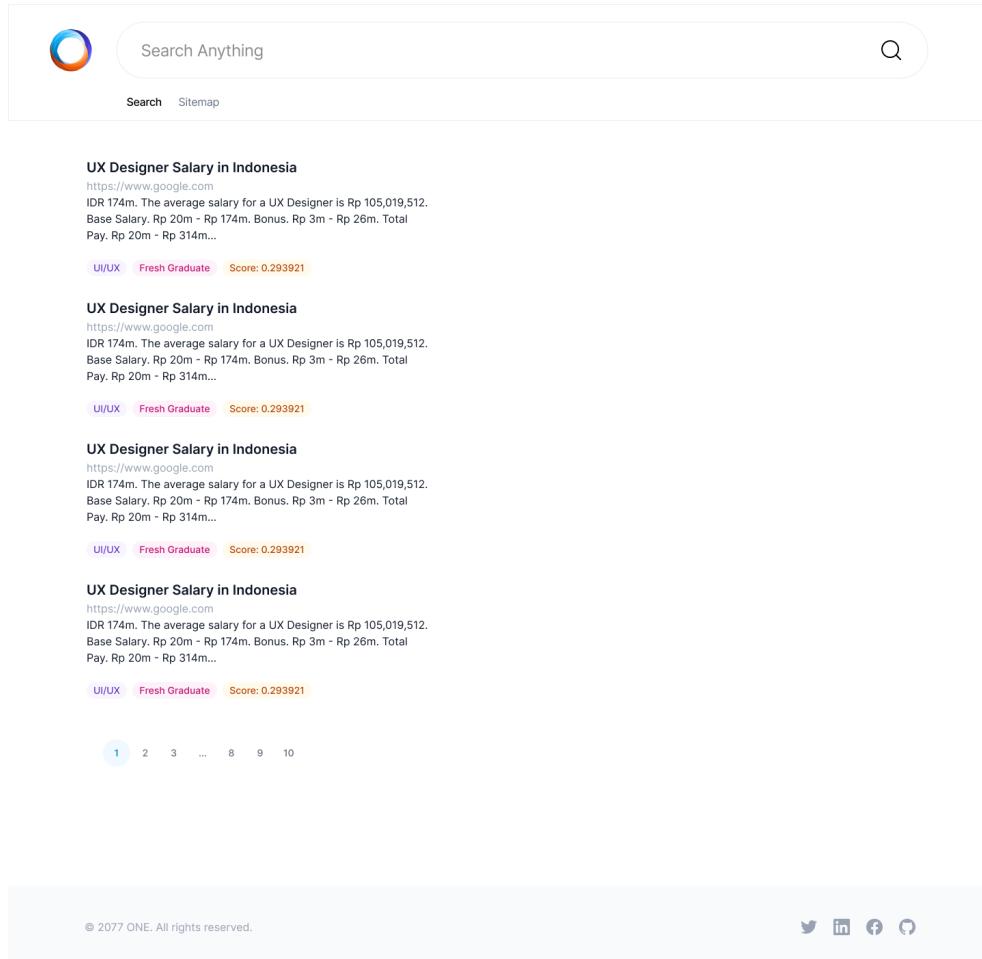
Pada halaman ini *staff* dapat mengubah informasi pribadi mereka.

- d). Merancang tampilan pencarian



Gambar 4.10: Rancangan tampilan pencarian

Pada halaman ini pengguna dapat melakukan pencarian halaman web yang mereka inginkan dengan memasukan kata kunci yang ingin pengguna cari ke dalam kolom yang telah disediakan lalu tekan tombol *enter*. Setelah itu pengguna akan dialihkan ke halaman hasil pencarian.



Gambar 4.11: Rancangan tampilan hasil pencarian

Pada halaman ini pengguna dapat melihat hasil pencarian yang mereka inginkan. Pada saat pengguna menekan salah satu tautan yang tersedia, pengguna akan dialihkan ke situs dalam *tab* baru. Pengguna juga dapat melihat hasil pencarian mereka dalam bentuk graf tiga dimensi dengan menekan tombol *sitemap*.

B. Sprint 2

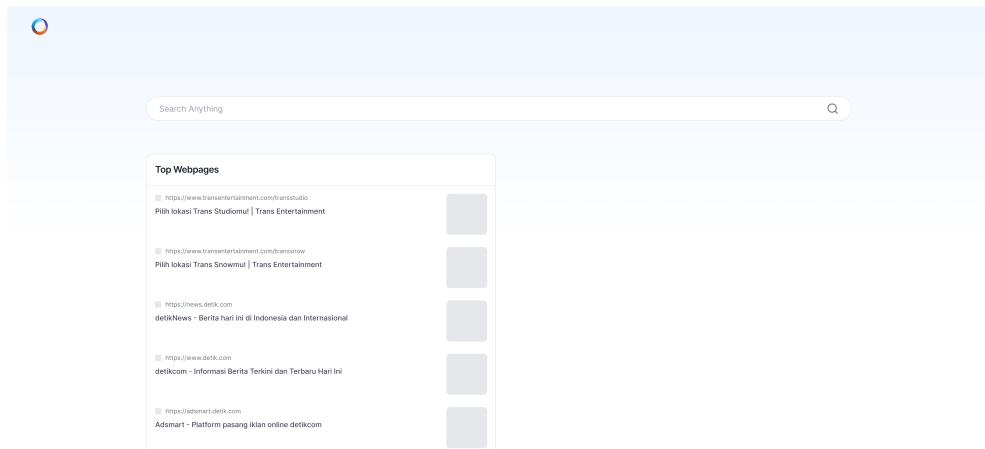
Tabel 4.3: Sprint 2 Backlog

No	Story	Task	Status
----	-------	------	--------

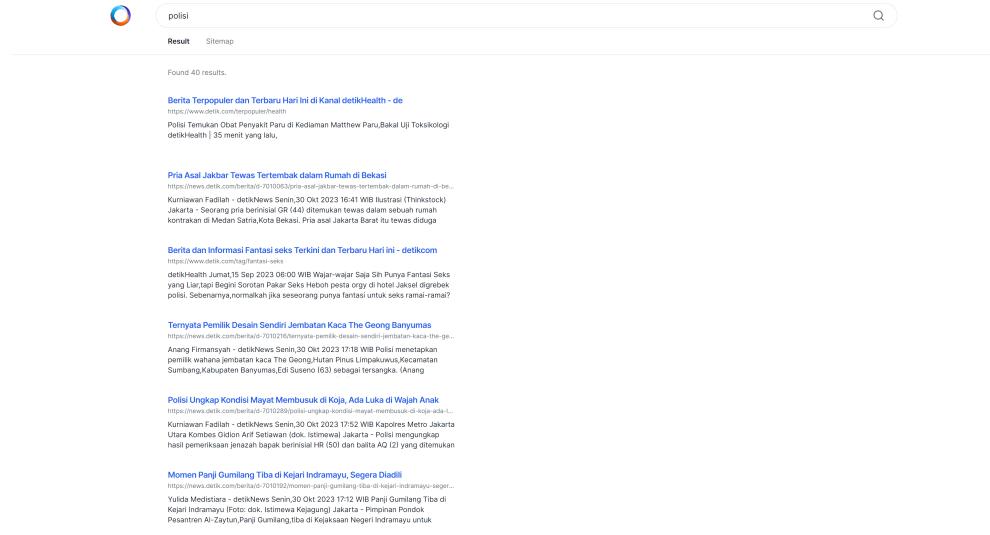
1	Fitur pencarian pengguna	Menerapkan <i>mock-up</i> tampilan halaman hasil pencarian dan pencarian <i>search engine</i>	selesai
		Membuat <i>mock-up</i> tampilan halaman hasil pencarian dan pencarian <i>search engine</i>	selesai (sprint 1)
		Membuat <i>Rest API</i> untuk tampilan halaman hasil pencarian dan pencarian <i>search engine</i>	belum
2	Struktur Projek	Merancang struktur projek yang akan digunakan	belum

a). Penerapan Tampilan Pencarian Pengguna

Pada tahap ini, dilakukan pengimplementasian *mock-up* tampilan yang telah dibuat pada sprint sebelumnya dengan menggunakan bahasa pemrograman *Javascript* dengan bantuan pustaka *Vue*.



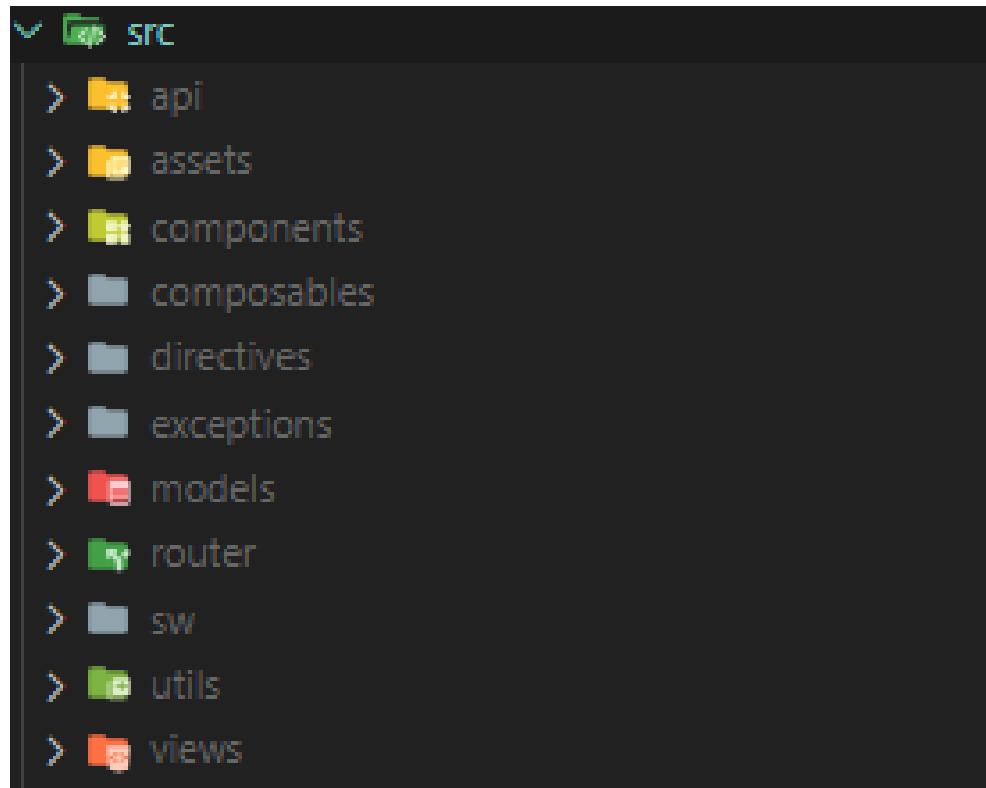
Gambar 4.12: Tampilan halaman pencarian *search engine*



Gambar 4.13: Tampilan halaman hasil pencarian *search engine*

b). Perancangan Struktur Kode Projek

Rancangan ini terdiri dari beberapa direktori utama yaitu *api*, *components*, *router* dan *views*. Direktori *api* berfungsi untuk menyimpan segala panggilan *web api* yang dilakukan oleh aplikasi. Direktori *components* berisi komponen-komponen tampilan aplikasi. Direktori *router* berisi dengan pemetaan tampilan aplikasi dan *views* merupakan direktori yang berfungsi sebagai penyimpanan berbagai tampilan layar aplikasi.



Gambar 4.14: Tampilan struktur kode projek *search engine*

c). *Unit Testing*

Tabel 4.4: *Unit Testing* Tampilan Pencarian

Skenario Pengujian	Sesuai	Tidak Sesuai
Pengguna dapat melakukan pencarian halaman web dengan memasukan kata kunci ke dalam <i>text input</i> yang tersedia	✓	
Pengguna dapat melihat hasil pencarian setelah melakukan pencarian	✓	
Pengguna dapat melihat hasil pencarian lebih banyak menggunakan <i>pagination</i> yang terletak pada bawah halaman pencarian	✓	

d). Perencanaan Sprint 3

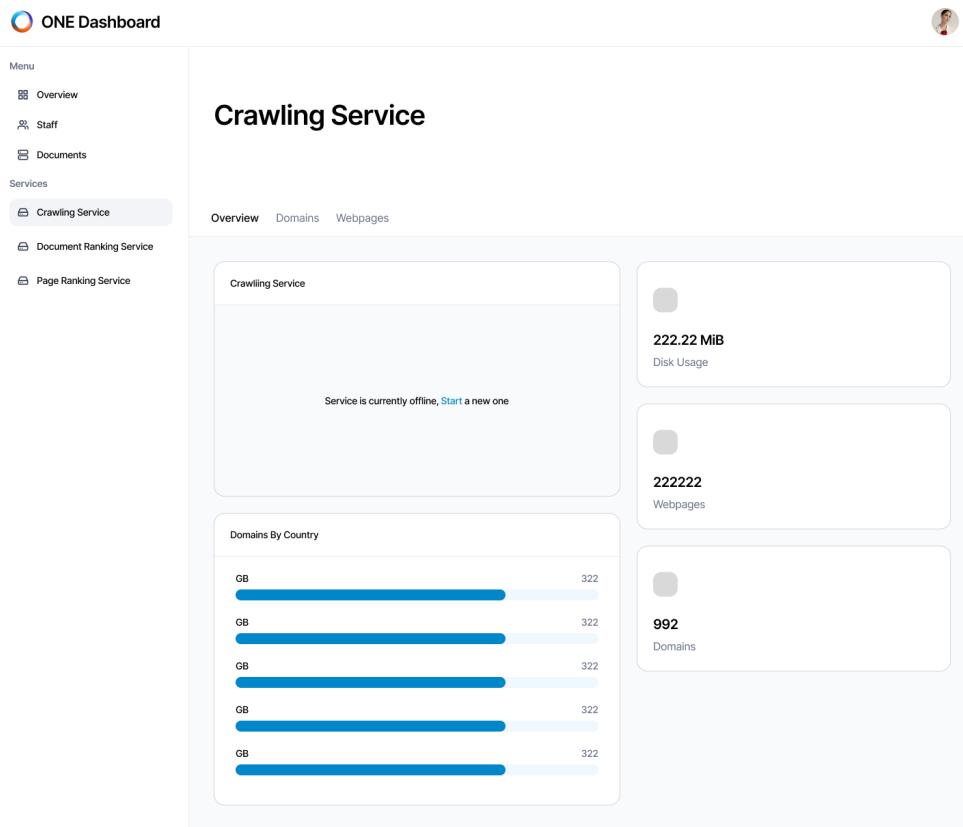
Pada akhir sprint diadakan perencanaan untuk *sprint 3*. Pada *sprint 3* akan dilakukan desain dan pengimplementasian desain halaman fitur *crawling*.

C. Sprint 3

Tabel 4.5: Sprint 3 Backlog

No	Story	Task	Status
1	Fitur crawling	Menerapkan <i>mock-up</i> tampilan status crawling <i>search engine</i>	selesai
		Membuat <i>mock-up</i> tampilan status crawling <i>search engine</i>	selesai
		Membuat <i>web service</i> untuk tampilan halaman status crawling <i>search engine</i>	belum

- a). Desain Tampilan Halaman Fitur Crawling



Gambar 4.15: Tampilan halaman *overview crawling search engine*

Halaman ini menampilkan ringkasan mengenai fitur *crawling*, pada halaman ini dapat menjalankan tugas *crawling* dengan menekan tombol *start*.

Halaman ini juga menyajikan informasi berupa jumlah penggunaan *storage*, jumlah halaman *web* dan jumlah *domains*.

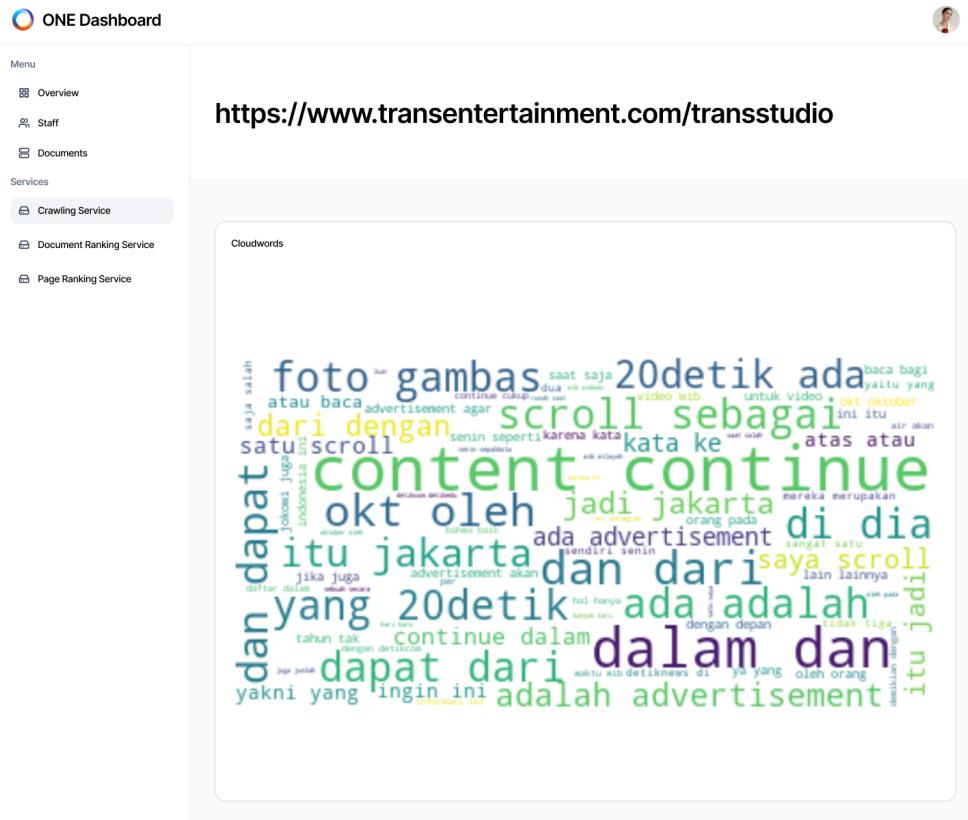
#	Name	Country	Pages
1	detik.com	ID	123312
1	detik.com	ID	123312
1	detik.com	ID	123312
1	detik.com	ID	123312
1	detik.com	ID	123312
1	detik.com	ID	123312
1	detik.com	ID	123312
1	detik.com	ID	123312
1	detik.com	ID	123312
1	detik.com	ID	123312

Gambar 4.16: Tampilan halaman *domains crawling search engine*

Halaman ini menyajikan daftar *domain* yang berhasil ter-*crawl*. Pada setiap *domain* terdapat informasi dari negara mana *domain* itu berasal dan jumlah halaman web yang ada dalam cakupan *domain* tersebut.

Gambar 4.17: Tampilan halaman hasil *crawling* halaman *web search engine*

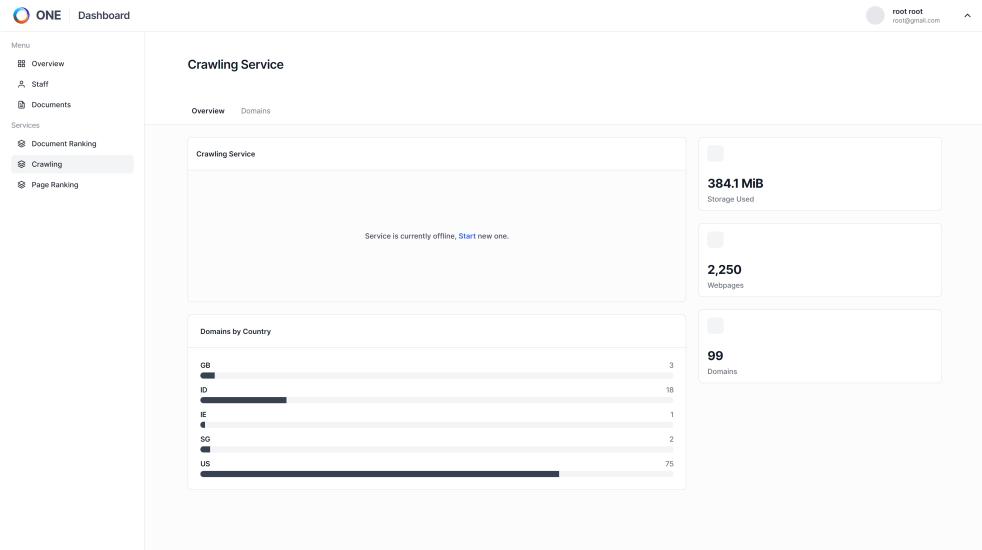
Halaman ini menyajikan daftar halaman *web* yang berhasil ter-*crawl*. Pada setiap halaman *web* terdapat informasi dari negara mana halaman *web* itu berasal, URL halaman *web* tersebut dan skor *pagerank*.



Gambar 4.18: Tampilan halaman detail halaman *web search engine*

b). Penerapan Tampilan Halaman Fitur Crawling

Pada tahap ini, dilakukan pengimplementasian *mock-up* tampilan yang telah dibuat dengan menggunakan bahasa pemrograman *Javascript* dengan bantuan pustaka *Vue*.



Gambar 4.19: Tampilan halaman status *crawling search engine*

#	Name	Country	Pages
1	detik.com	ID	311
2	linkedin.com	US	226
3	20.detik.com	ID	149
4	news.detik.com	ID	131
5	sport.detik.com	ID	117
6	thinkwithgoogle.com	US	114
7	telegraph.co.uk	US	94
8	finance.detik.com	ID	89
9	inet.detik.com	ID	81
10	google.com	US	80
11	oto.detik.com	ID	79
12	blog.google	US	77
13	food.detik.com	ID	69
14	health.detik.com	ID	66
15	wajipop.detik.com	ID	62

Gambar 4.20: Tampilan halaman daftar domain *crawling search engine*

c). Unit Testing

Tabel 4.6: Unit Testing Tampilan Crawling

Skenario Pengujian	Sesuai	Tidak Sesuai
--------------------	--------	--------------

Pengguna dapat memulai proses <i>crawling</i> dengan mengklik tulisan <i>start</i>	✓	
Pengguna dapat menghentikan jalannya <i>crawling</i> dengan mengklik tulisan <i>stop</i>	✓	
Saat <i>tab domains</i> diklik, pengguna akan dialihkan ke sub halaman daftar domain	✓	
Saat <i>tab webpages</i> diklik, pengguna akan dialihkan ke sub halaman daftar webpage	✓	

d). Pengujian Sprint 3 dan Perencanaan Sprint 4

Pada akhir sprint ini telah dilakukan *pengujian* mengenai fitur yang telah dibuat dengan *stakeholder*. Tidak ditemukan adanya kendala. Pada perencanaan *Sprint 4* akan dilakukan pengimplementasian desain *page ranking*.

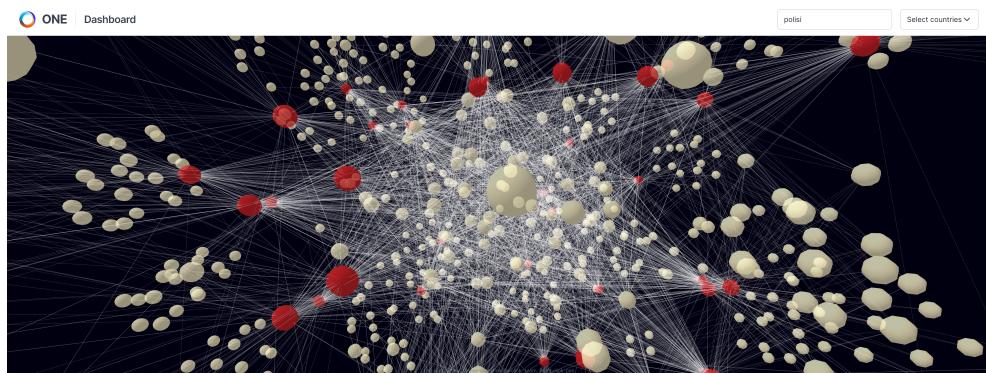
D. *Sprint 4*

Tabel 4.7: Sprint 4 Backlog

No	Story	Task	Status
1	Fitur <i>page ranking</i>	Menerapkan <i>mock-up</i> tampilan status <i>page ranking search engine</i>	belum
		Membuat <i>mock-up</i> tampilan status <i>page ranking search engine</i>	belum
		Membuat <i>web service</i> untuk tampilan halaman <i>page ranking search engine</i>	belum
		Membuat <i>mock-up</i> tampilan peta situs <i>search engine</i>	selesai (sprint 1)

	Menerapkan <i>mockup</i> tampilan peta situs	selesai
--	--	---------

a). Penerapan Tampilan Peta Situs



Gambar 4.21: Tampilan halaman peta situs

b). *Unit Testing*

Tabel 4.8: *Unit Testing* Tampilan Page Ranking

Skenario Pengujian	Sesuai	Tidak Sesuai
Pengguna dapat mem-filter peta situs berdasarkan kata kunci dan negara asal situs	✓	

c). Pengujian Sprint 4 dan Perencanaan Sprint 5

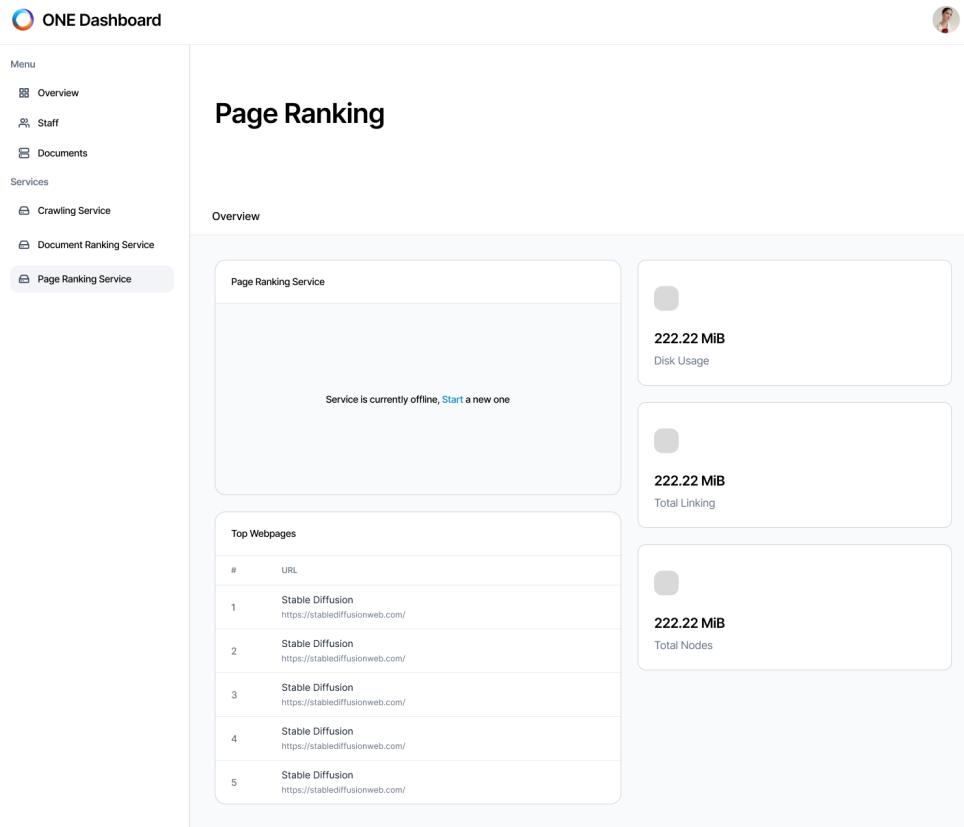
Pada akhir sprint ini telah dilakukan *pengujian* mengenai fitur yang telah dibuat dengan *stakeholder*. Ditemukan kendala pada penerapan peta situs dimana peta situs untuk semua yang dibuat terasa berat dari sisi performa, sehingga peta situs disederhanakan menjadi hanya menampilkan peta situs setelah pengguna memasukan kata kunci pencarian. Pada perencanaan *Sprint 5* akan dilakukan pembuatan lalu pengimplementasian desain *page ranking* dan pembuatan lalu pengimplementasian desain *document ranking*.

E. *Sprint 5*

Tabel 4.9: Sprint 5 Backlog

No	Story	Task	Status
1	Fitur <i>page ranking</i>	Menerapkan <i>mock-up</i> tampilan status <i>page ranking search engine</i>	selesai
		Membuat <i>mock-up</i> tampilan peta situs <i>search engine</i>	selesai (sprint 4)
		Membuat <i>mock-up</i> tampilan status <i>page ranking search engine</i>	selesai
		Membuat <i>web service</i> untuk tampilan halaman <i>page ranking search engine</i>	belum
2	Fitur <i>staff</i>	Menerapkan tampilan login, daftar, tambah staff	selesai
		Membuat <i>mock-up</i> tampilan login, daftar, tambah staff	selesai (sprint 1)
		Membuat <i>web service</i> untuk tampilan login, daftar, tambah staff <i>search engine</i>	belum
3	Fitur <i>document ranking</i>	Menerapkan <i>mock-up</i> tampilan status <i>document ranking search engine</i>	selesai
		Membuat <i>mock-up</i> tampilan status <i>document ranking search engine</i>	selesai
		Membuat <i>web service</i> untuk tampilan halaman <i>document ranking search engine</i>	belum

- a). Pembuatan *mockup* tampilan *page ranking*



Gambar 4.22: Tampilan halaman *overview page ranking search engine*

Halaman ini menampilkan ringkasan mengenai fitur *page ranking*, pada halaman ini dapat menjalankan tugas *page ranking* dengan menekan tombol *start*. Halaman ini juga menyajikan informasi berupa jumlah penggunaan *storage*, jumlah koneksi antara halaman *web*, jumlah halaman *web*, jumlah halaman *web* yang ada dalam *database*, *wordcloud* dan kata kunci yang paling banyak dicari.

- Pembuatan *mockup* tampilan *document ranking*

Document Ranking

Overview Search Log Words

#	Keyword	Frequency
1	Reguler	2222
2	Reguler	2222
3	Reguler	2222
4	Reguler	2222
5	Reguler	2222

Cloudwords

Cloudwords visualization showing frequently searched keywords in a cloud format.

Gambar 4.23: Tampilan halaman *overview document ranking search engine*

Halaman ini menampilkan ringkasan mengenai fitur *document ranking*, pada halaman ini dapat menjalankan tugas *document ranking* dengan menekan tombol *start*. Halaman ini juga menyajikan informasi berupa kata kata unik, jumlah halaman *web* yang ada dalam *database*, *wordcloud* dan kata kunci yang paling banyak dicari.

The screenshot shows the ONE Dashboard interface. On the left, there's a sidebar with a menu containing 'Overview', 'Staff', 'Documents', 'Services', 'Crawling Service', 'Document Ranking Service' (which is selected and highlighted in blue), and 'Page Ranking Service'. The main content area has a title 'Document Ranking' and tabs for 'Overview', 'Search Log' (which is active and highlighted in blue), and 'Words'. Below the tabs is a search bar with placeholder 'Search' and a dropdown for 'Per Page: 25'. A table follows, displaying 10 rows of search log data. The columns are '#', 'Keyword', 'IP Address', 'User Agent', and 'Date'. The data shows multiple entries for the keyword 'Motor Listrik' from IP address 103.22.412.33, all occurring on August 28, 2023, at 07:02:10. The user agent for these entries is Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4453.113 Safari/537.36. The table also includes navigation links at the bottom: '0 - 20 of 67', 'Previous', and 'Next'.

#	Keyword	IP Address	User Agent	Date
1	2024	103.22.412.33	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4453.113 Safari/537.36 (28 August 2023 07:02:10
2	Motor Listrik	103.22.412.33	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4453.113 Safari/537.36 (28 August 2023 07:02:10
3	Motor Listrik	103.22.412.33	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4453.113 Safari/537.36 (28 August 2023 07:02:10
4	Motor Listrik	103.22.412.33	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4453.113 Safari/537.36 (28 August 2023 07:02:10
5	Motor Listrik	103.22.412.33	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4453.113 Safari/537.36 (28 August 2023 07:02:10
6	Motor Listrik	103.22.412.33	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4453.113 Safari/537.36 (28 August 2023 07:02:10
7	Motor Listrik	103.22.412.33	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4453.113 Safari/537.36 (28 August 2023 07:02:10
8	Motor Listrik	103.22.412.33	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4453.113 Safari/537.36 (28 August 2023 07:02:10
9	Motor Listrik	103.22.412.33	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4453.113 Safari/537.36 (28 August 2023 07:02:10
10	Motor Listrik	103.22.412.33	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4453.113 Safari/537.36 (28 August 2023 07:02:10

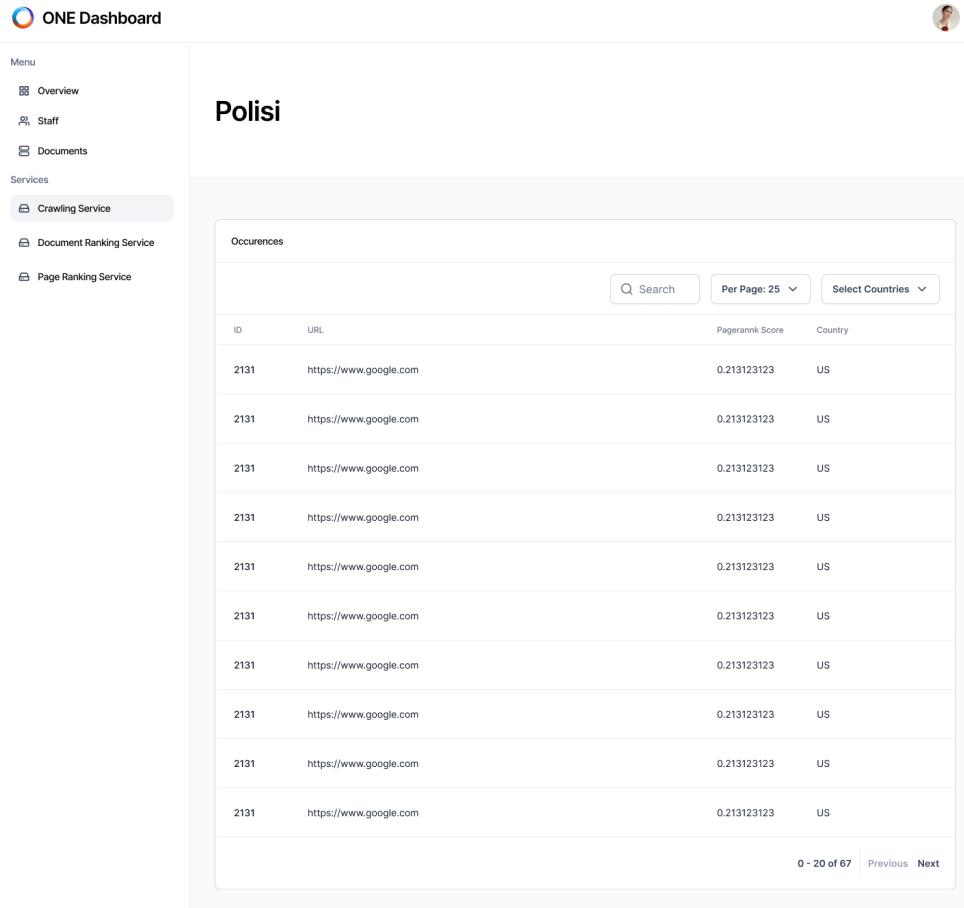
Gambar 4.24: Tampilan halaman *overview document ranking search logs*

Pada halaman *search logs* terdapat daftar kata-kata yang pernah pengguna cari dengan *search engine*. Data yang dikumpulkan berupa halaman *web*, kata kunci, *browser* yang pengguna gunakan dan tanggal.

#	Word	Frequency
1	2024	103.22.412.33
2	Motor Listrik	103.22.412.33
3	Motor Listrik	103.22.412.33
4	Motor Listrik	103.22.412.33
5	Motor Listrik	103.22.412.33
6	Motor Listrik	103.22.412.33
7	Motor Listrik	103.22.412.33
8	Motor Listrik	103.22.412.33
9	Motor Listrik	103.22.412.33
10	Motor Listrik	103.22.412.33

Gambar 4.25: Tampilan halaman daftar kata *document ranking*

Halaman ini menyajikan daftar kata-kata yang terdapat dalam *database* diurutkan mulai dari jumlah penggunaan. Pada saat pengguna menekan salah satu kata maka pengguna akan diantarkan ke halaman detail kata.



Gambar 4.26: Tampilan halaman detail kata *document ranking*

Pada halaman detail kata, disajikan sebuah daftar halaman *web* yang mengandung kata tersebut.

- c). Penerapan tampilan *page ranking search engine*

The screenshot shows the 'Page Ranking' service overview. It includes a summary section with 'Disk Usage' (384.1 MiB) and 'Total Linking' (337,251). Below this is a table of 'Top Webpages' with the following data:

Rank	URL
1	Pilih Isak! Trans StudioMul Trans Entertainment http://www.transentertainment.com/transisak
2	Pilih Isak! Trans Snowmul Trans Entertainment http://www.transentertainment.com/transsnow
3	detikNews - Berita hari ini di Indonesia dan Internasional https://news.detik.com
4	detikcom - Informasi Berita Terkini dan Terbaru Hari Ini https://www.detik.com
5	Adsmart - Platform pasang iklan online detikcom https://adsmart.detik.com
6	Detik Kariir - Daftar Lowongan Kerja Terbaru di Detikcom https://kariir.detik.com
7	detikX - Informasi Mendalam dan Interaktif https://www.detik.com/x
n	CXO Media Welcome to CXO Media

Gambar 4.27: Tampilan halaman *overview page ranking search engine*

d). Penerapan tampilan *staff*

The screenshot shows the 'Add Staff' form. It has two main sections: 'Personal Info' and 'Role'. The 'Personal Info' section contains fields for Email, Password, First Name, and Last Name, each with an 'Enter value' placeholder. The 'Role' section is a dropdown menu with a placeholder 'Enter value'. At the bottom right is a 'Create' button.

Gambar 4.28: Tampilan halaman tambah *staff*

The screenshot shows the 'Staff' list page in the ONE application. The left sidebar contains a 'Menu' with items: Overview, Staff (which is selected), Documents, Services, Document Ranking, Crawling, and Page Ranking. The main content area has a header 'Staff' with a 'Create New Staff' button. Below is a table with columns: ID, Email, Full Name, and Role. Two staff entries are listed:

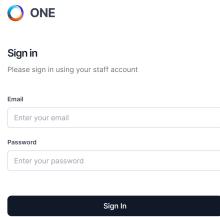
ID	Email	Full Name	Role
4	root@gmail.com	root root	root
5	rooxt@gmail.com	asd asd	staff

At the bottom right, there are links for '0 - 2 of 2', 'Previous', and 'Next'.

Gambar 4.29: Tampilan halaman daftar *staff*

The screenshot shows the 'Update Profile' page in the ONE application. The left sidebar is identical to the previous screenshot. The main content area has a header 'Update Profile' with a 'Personal Info' section. It includes fields for Email (root@gmail.com), Password (Enter password), First Name (root), and Last Name (root). At the bottom right is a 'Update' button.

Gambar 4.30: Tampilan halaman edit *profile*



Gambar 4.31: Tampilan halaman *login staff*

e). *Unit Testing*

Tabel 4.10: *Unit Testing* Tampilan Staff

Skenario Pengujian	Sesuai	Tidak Sesuai
Staff yang bertindak sebagai Root dapat menambahkan staff baru dengan menekan tombol <i>create</i> dan mengisi formulir yang disajikan	✓	
Staff yang bertindak sebagai Root dapat menghapus staff dengan menekan tombol <i>delete</i>	✓	
Staff dapat meng- <i>edit profile</i> mereka dengan mengisi form yang disajikan	✓	

Tabel 4.11: *Unit Testing* Tampilan Page Ranking

Skenario Pengujian	Sesuai	Tidak Sesuai
--------------------	--------	--------------

Pengguna dapat menekan tombol <i>start</i> untuk memulai <i>page ranking</i>	✓	
--	---	--

Tabel 4.12: *Unit Testing* Tampilan Document Ranking

Skenario Pengujian	Sesuai	Tidak Sesuai
Pengguna dapat menekan tombol <i>start</i> untuk memulai <i>document ranking</i>	✓	
Pengguna dapat menekan tombol <i>stop</i> untuk memulai <i>document ranking</i>	✓	
Saat <i>tab search log</i> diklik maka pengguna akan dialihkan ke sub halaman <i>search log</i>	✓	
Saat <i>tab words</i> diklik maka pengguna akan dialihkan ke sub halaman <i>words</i>	✓	

F. Sprint 6**Tabel 4.13:** *Sprint 3 Backlog*

No	Story	Task	Status
1	Fitur crawling	Menerapkan <i>mock-up</i> tampilan status crawling <i>search engine</i>	selesai (sprint 3)
		Membuat <i>mock-up</i> tampilan status crawling <i>search engine</i>	selesai (sprint 3)
		Membuat <i>web service</i> untuk tampilan halaman status crawling <i>search engine</i>	selesai

- a). Perancangan *Rest API*

The screenshot shows the Postman interface with the following details:

- Request URL:** GET {{host}}/{{prefix}}/crawling/status
- Params:** host (host: http://localhost:8080)
- Body (Pretty):**

```

1 {
2   "status": "IDLE"
3 }
```
- Status:** 200 OK
- Activate Windows:** Go to Settings to activate Windows.

Gambar 4.32: REST API crawling status

The screenshot shows the Postman interface with the following details:

- Request URL:** GET {{host}}/{{prefix}}/crawling/metrics
- Params:** None
- Body (Pretty):**

```

1 {
2   "countries": [...]
3 ],
4   "domains_stats": [
5     {
6       "total_domains": 99,
7       "total_webpages": 2250,
8       "total_webpages_size": 402767965
9     }
10 }
```
- Status:** 200 OK
- Activate Windows:** Go to Settings to activate Windows.

Gambar 4.33: REST API crawling metrics

The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** `((host))/((prefix))/document_ranking/start?duration=6000&threads=3`
- Params:** duration: 6000, threads: 3
- Body:** JSON response:


```

1
2   "message": "started!"
3
      
```
- Status:** 200 OK
- Message:** Activate Windows. Go to Settings to activate Windows.

Gambar 4.34: REST API crawling start

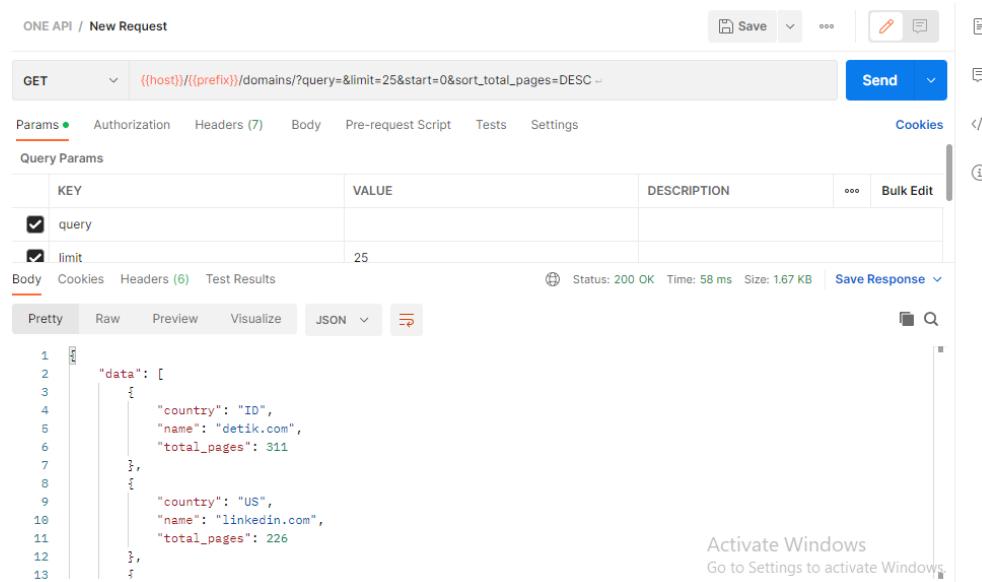
The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** `((host))/((prefix))/document_ranking/stop`
- Params:** Key
- Body:** JSON response:


```

1
2   "message": "nothing running!"
3
      
```
- Status:** 200 OK
- Message:** Activate Windows. Go to Settings to activate Windows.

Gambar 4.35: REST API crawling stop



Gambar 4.36: REST API domains

b). Pengujian Sprint 6

Pada akhir sprint ini telah dilakukan *pengujian* mengenai fitur yang telah dibuat dengan *stakeholder*. Ditemukan kendala berupa tampilan *domains by country*. Pada rancangan tampilan *domains by country* hanya dijabarkan jumlah domain dan negara saja tanpa adanya pengelompokan negara, *stakeholder* menyarankan untuk mengelompokan beberapa negara ke dalam kelompok tertentu seperti Indonesia, Asia Tenggara, Asia Timur (Jepang, Korea, Taiwan), Asia Selatan (Bangladesh, India, Pakistan), Timur Tengah (semua), China, Russia, Eropa (semua), Afrika, Amerika Utara, Amerika Selatan, Asia Pasifik (New Guinea, negara-negara kecil di pasifik), Australia.

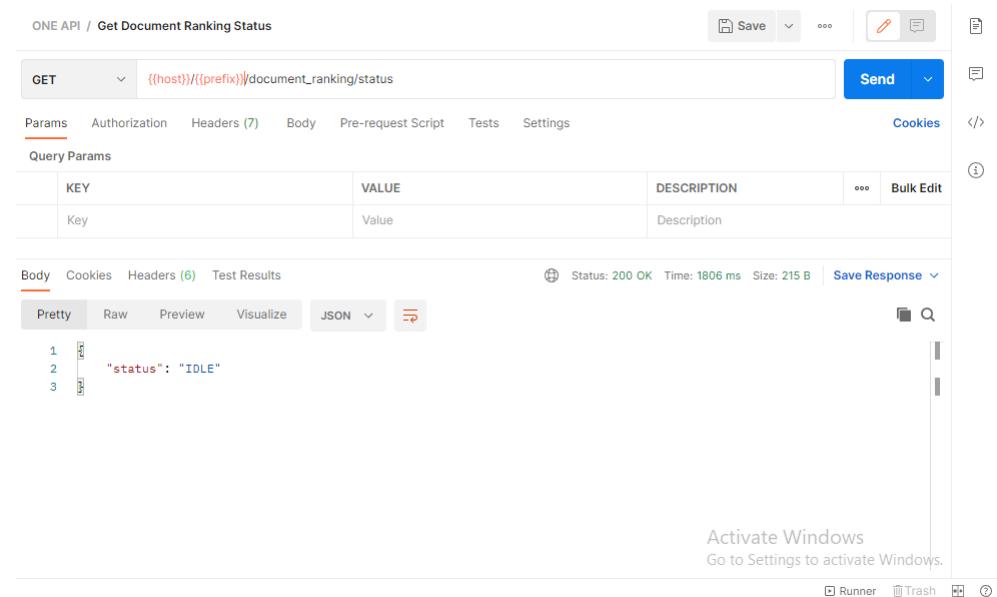
G. *Sprint 7*

Tabel 4.14: Sprint 7 Backlog

No	Story	Task	Status
1	Fitur <i>document ranking</i>	Menerapkan <i>mock-up</i> tampilan status <i>document ranking search engine</i>	selesai (sprint 5)

	Membuat <i>mock-up</i> tampilan status <i>document ranking search engine</i>	selesai (sprint 5)
	Membuat <i>Rest API document ranking</i>	selesai

a). Perancangan *Rest API*



The screenshot shows the Postman application interface. At the top, it displays the URL: `GET {{host}}/({prefix})/document_ranking/status`. Below the URL, there are tabs for Params, Authorization, Headers (7), Body, Pre-request Script, Tests, and Settings. The Params tab is selected, showing a single entry: Key under KEY and Value under VALUE. The Body tab is also selected, showing the JSON response:

```
1 "status": "IDLE"
```

. The response status is 200 OK with a time of 1806 ms and a size of 215 B. The interface includes various buttons like Save, Send, and a sidebar with file management options.

Gambar 4.37: REST API status document ranking

The screenshot shows the Postman interface with the following details:

- URL:** {{host}}/{{prefix}}/document_ranking/start?algorithm=inverted-indexer&use_gst=true
- Method:** POST
- Params:**
 - algorithm: inverted-indexer
 - use_gst: true
- Body:** (Pretty, Raw, Preview, Visualize, JSON) - Shows a JSON response with the key "message": "started!"
- Status:** 200 OK, Time: 1979 ms, Size: 220 B
- Annotations:** "Activate Windows Go to Settings to activate Windows."

Gambar 4.38: REST API start document ranking

The screenshot shows the Postman interface with the following details:

- URL:** {{host}}/{{prefix}}/document_ranking/stop
- Method:** POST
- Params:**
 - Key: Value
- Body:** (Pretty, Raw, Preview, Visualize, JSON) - Shows a JSON response with the key "message": "nothing running!"
- Status:** 200 OK, Time: 2.02 s, Size: 228 B
- Annotations:** "Activate Windows Go to Settings to activate Windows."

Gambar 4.39: REST API stop document ranking

ONE API / Get Top Searched Keywords

GET `({host})/({prefix})/analytics/top_searched_words?start=0&limit=10&query=`

Params	Authorization	Headers (7)	Body	Pre-request Script	Tests	Settings	Cookies
KEY	VALUE	DESCRIPTION	...	Bulk Edit			
<input checked="" type="checkbox"/> start	0						
<input checked="" type="checkbox"/> limit	10						
<input checked="" type="checkbox"/> query							

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON

```

1
2   "data": [
3     {
4       "frequency": 38,
5       "query": "polisi"
6     },
7     {
8       "frequency": 6,
9       "query": "trans"
10    },
11    {
12      "frequency": 6,
13      "query": "detik"
14    }
15  ]

```

Status: 200 OK Time: 66 ms Size: 495 B Save Response

Activate Windows
Go to Settings to activate Windows

Runner Trash

Gambar 4.40: REST API kata paling banyak dicari

ONE API / Get Words

GET `({host})/({prefix})/words/?start=0&limit=25`

Params	Authorization	Headers (7)	Body	Pre-request Script	Tests	Settings	Cookies
Query Params	KEY	VALUE	DESCRIPTION	...	Bulk Edit		
<input checked="" type="checkbox"/> start	0						
<input checked="" type="checkbox"/> limit	25						

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON

```

1
2   "data": [
3     {
4       "frequency": 857,
5       "id_word": 1685527,
6       "page_id": 11371,
7       "tfidf_score": 0.17167857194199823,
8       "word": "2023"
9     },
10    {
11      "frequency": 806,
12      "id_word": 1685483,
13      "page_id": 11365,
14    }
15  ]

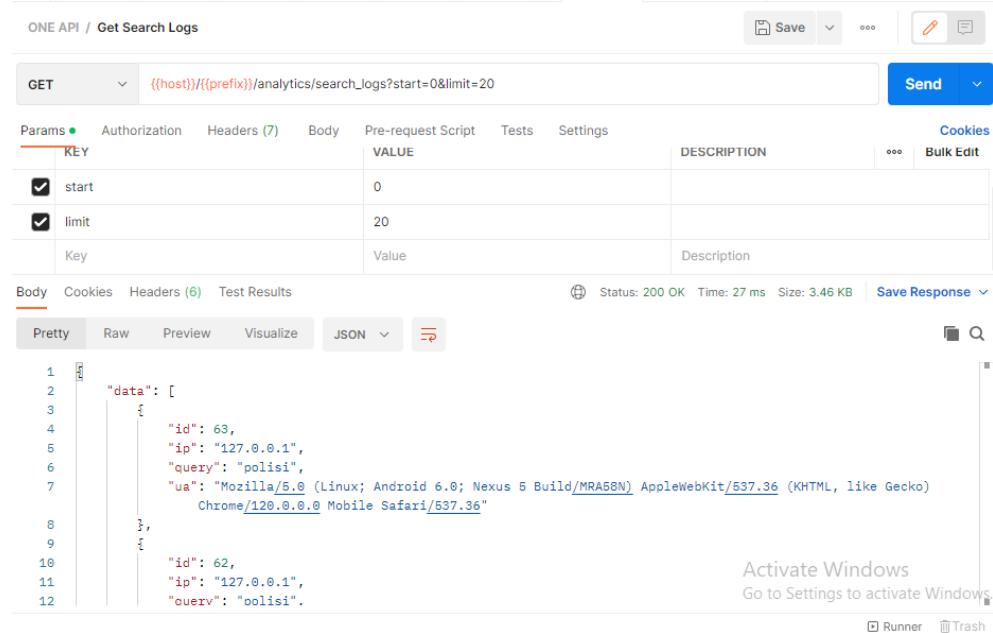
```

Status: 200 OK Time: 3.64 s Size: 2.73 KB Save Response

Activate Windows
Go to Settings to activate Windows

Runner Trash

Gambar 4.41: REST API words

**Gambar 4.42:** REST API search logs

H. Sprint 8

Tabel 4.15: Sprint 8 Backlog

No	Story	Task	Status
1	Fitur pencarian pengguna	Menerapkan <i>mock-up</i> tampilan halaman hasil pencarian dan pencarian <i>search engine</i>	selesai (sprint 2)
		Membuat <i>mock-up</i> tampilan halaman hasil pencarian dan pencarian <i>search engine</i>	selesai (sprint 1)
		Membuat Rest API untuk tampilan halaman hasil pencarian dan pencarian <i>search engine</i>	selesai
2	Fitur staff	Menerapkan tampilan login, daftar, tambah staff	selesai (sprint 5)
		Membuat <i>mock-up</i> tampilan login, daftar, tambah staff	selesai (sprint 1)

		Membuat <i>Rest API</i> untuk tampilan login, daftar, tambah staff, <i>edit profile search engine</i>	selesai
3	Fitur <i>page ranking</i>	Menerapkan <i>mock-up</i> tampilan status <i>page ranking search engine</i>	selesai (sprint 5)
		Membuat <i>mock-up</i> tampilan status <i>page ranking search engine</i>	selesai (sprint 2)
		Membuat <i>Rest API</i> untuk tampilan halaman <i>page ranking search engine</i>	selesai

ONE API / Get Webpages

GET

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

<input checked="" type="checkbox"/> sort_pagerank_score	DESC
<input checked="" type="checkbox"/> start	0
<input type="checkbox"/> query	
<input checked="" type="checkbox"/> countries[]	IE
<input checked="" type="checkbox"/> limit	200

Body Cookies Headers (6) Test Results Status: 200 OK Time: 185 ms Size: 2.26 KB Save Response

Pretty Raw Preview Visualize JSON

```

2   "data": [
3     {
4       "country": "IE",
5       "id": 11696,
6       "keywords": [],
7       "pagerank_score": 0.0009333416196571743,
8       "title": "detikcom",
9       "total_words": 1,
10      "url": "https://www.facebook.com/detikcom"
    
```

Activate Windows
Go to Settings to activate Windows

Runner Trash

Gambar 4.43: REST API untuk mendapatkan halaman web

The screenshot shows a Postman request for a 3D Sitemap. The URL is `https://{{host}}/{{prefix}}/sitemap/3d?query=polisi`. The 'Params' tab is selected, showing a query parameter 'query' with the value 'polisi'. The 'Body' tab is selected, showing a JSON response with a 'links' array containing two objects. Each object has a 'source' and a 'target' URL, both linking to news articles from detik.com. The status is 200 OK with a response size of 1.13 MB.

```

1
2   "links": [
3     {
4       "source": "https://news.detik.com/berita/d-7009580/
5         _dugaan-korupsi-kementerian-ternyata-dilaporkan-ke-kpk-sejak-februari-2020",
6       "target": "https://news.detik.com/berita/d-7009580/
7         _dugaan-korupsi-kementerian-ternyata-dilaporkan-ke-kpk-sejak-februari-2020"
8     },
9   ]
  
```

Gambar 4.44: REST API untuk peta situs

The screenshot shows a Postman request for Overall Similarity. The URL is `https://{{host}}/{{prefix}}/overall_ranking/similarity?length=25&start=0&keyword=polisi`. The 'Params' tab is selected, showing 'length' set to 25 and 'start' set to 0. The 'Body' tab is selected, showing a JSON response with a 'data' array containing one object. This object contains details about a news article from detikHealth. The status is 200 OK with a response size of 67.77 KB.

```

1
2   "data": [
3     {
4       "content_text": "Polisi Temukan Obat Penyakit Paru di Kediaman Matthew Paru, Bakal Uji Toksikologi
5         detikHealth | 35 menit yang lalu",
6       "country": "ID",
7       "description": "Rangkuman Berita Terpopuler dan Terbaru Hari Ini di Kanal detikHealth - detikcom",
8       "id_page": 11552,
9       "pagerank_score": 0.00018819578262659802,
10      "similarity_score": 0.12846960918842315,
11      "tfidf_total": 0.21399055145895415,
12      "title": "Berita Terpopuler dan Terbaru Hari Ini di Kanal detikHealth - de",
13      "url": "https://www.detik.com/terpopuler/health"
  
```

Gambar 4.45: REST API untuk halaman hasil pencarian

The screenshot shows a POST request to `({host})/({prefix})/auth/login`. The request body is JSON with fields `email: "ggg@gmail.com"` and `password: "gggggggg"`. The response status is 400 BAD REQUEST, with the message `"message": "Account not found"`.

Gambar 4.46: REST API untuk login

The screenshot shows a GET request to `({host})/({prefix})/page_ranking/status`. The query parameter `status` is set to `IDLE`. The response status is 200 OK, with the message `"status": "IDLE"`.

Gambar 4.47: REST API untuk status page ranking

The screenshot shows a Postman request for a 'page_ranking/stop' endpoint. The status is 500 INTERNAL SERVER ERROR. The response body is:

```

1
2   "message": "no process are running in",
3   "ok": false
4

```

Gambar 4.48: REST API untuk stop page ranking

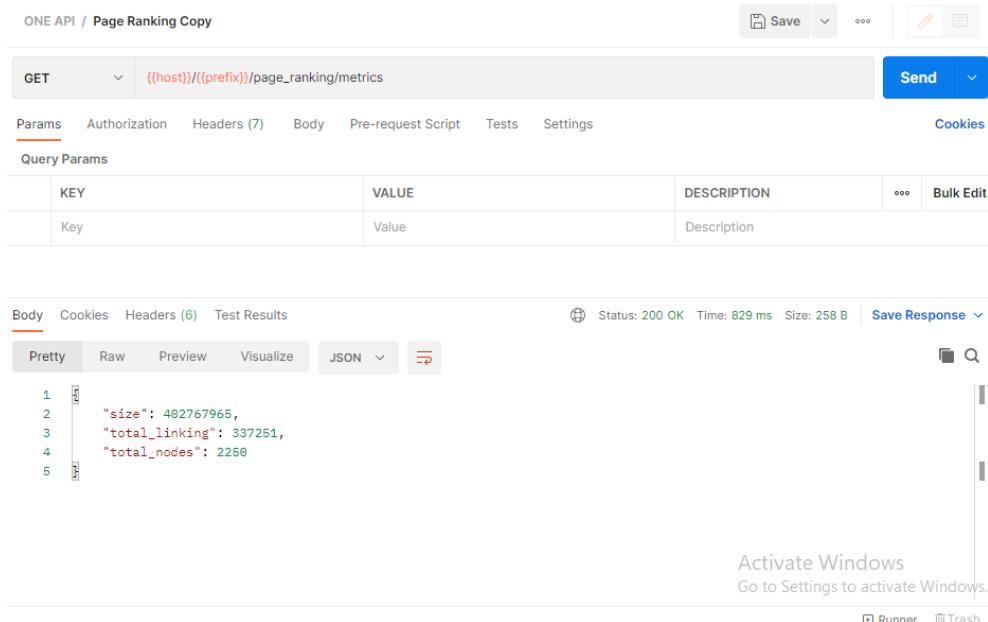
The screenshot shows a Postman request for a 'page_ranking/start' endpoint. The status is 200 OK. The response body is:

```

1
2   "message": "Sukses",
3   "ok": true
4

```

Gambar 4.49: REST API untuk start page ranking



Gambar 4.50: REST API untuk *page ranking metrics*

I. Sprint 9

Tabel 4.16: Sprint 10 Backlog

No	Story	Task	Status
1	Fitur <i>multi-threaded service</i>	Menerapkan <i>multi-threaded</i> pada <i>service page ranking</i>	selesai
		Menerapkan <i>multi-threaded</i> pada <i>service document ranking</i>	selesai

Penerapan *multi-threaded* pada aplikasi terutama pada bagian *page ranking* dan *document ranking* bertujuan untuk memanfaatkan sumber daya komputasi *server* secara optimal. Penerapan ini menggunakan *API ThreadPoolExecutor* yang berasal dari pustaka *concurrent* dengan bahasa pemrograman *python*. Adapun potongan kode dari penerapan *multi-threaded* pada *page ranking* sebagai berikut.

```

1 pages = get_all_crawled_pages(db_connection)
2 with ThreadPoolExecutor(16) as executor:
3     for result, t in executor.map(process_page, pages):
4         pr_change_sum += result

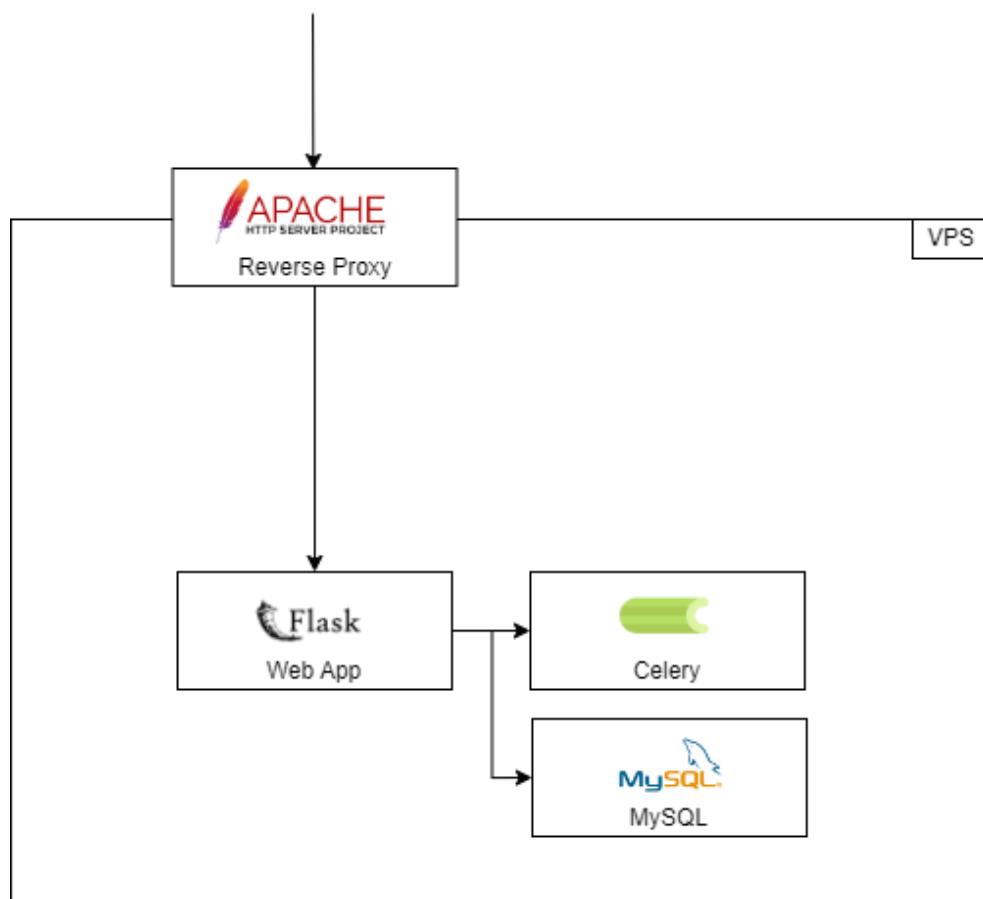
```

J. Sprint 10

Tabel 4.17: Sprint 11 Backlog

No	Story	Task	Status
1	<i>Service Deployment</i>	Men-deploy search engine pada server	selesai

Dalam *sprint 11* ini dirancang suatu infrastruktur web untuk mendukung aplikasi yang sudah dibuat, infrastruktur ini bertujuan untuk agar publik dapat mengakses aplikasi *web* yang telah dibuat.



Gambar 4.51: Desain infrastruktur *search engine*

Pada infrastruktur ini, Aplikasi *Apache* bertujuan untuk meneruskan permintaan dari internet publik ke aplikasi web yang terdapat dalam *Server* yaitu aplikasi web berbasis *Flask* dengan bahasa pemrograman *Python*. Aplikasi web

berbasis *flask* ini menangani seluruh permintaan yang datang dari pengguna dan mengirim respon yang sesuai dengan permintaan pengguna. Saat menjalankan tugasnya, aplikasi berbasis *flask* ini didampingi oleh beberapa aplikasi lainnya yaitu *Celery* dan *MySQL*. Aplikasi *Celery* digunakan oleh aplikasi web berbasis *Flask* untuk memindahkan sebagian beban kerja yang besar dari aplikasi web berbasis *flask* yang ada. Dengan adanya pemindahan beban dari aplikasi web berbasis *flask* ke aplikasi *Celery* ini, aplikasi web berbasis *flask* dapat melayani banyak permintaan pengguna yang masuk. Aplikasi *MySQL* merupakan aplikasi basis data yang digunakan dalam aplikasi web berbasis *Flask*. Aplikasi *MySQL* digunakan untuk menyimpan informasi dan mengambil data yang digunakan oleh aplikasi berbasis *Flask* yang ada.

Perilisan aplikasi pada *server* dimulai dengan menambahkan barisan kode pada aturan aplikasi *Apache* sebagai berikut

```

1 <VirtualHost *:80>
2 ...
3 WSGIDaemonProcess /se2 python-path=/opt/rh/rh-python38/root64/lib
4           /python3.8/site-packages
5 WSGIProcessGroup /se2
6 WSGIAApplicationGroup %{GLOBAL}
7 WSGIScriptAlias /se2 /var/www/html/se2/search-engine.wsgi
8 WSGIScriptReloading on
9 <Directory "/var/www/html/se2/src">
10 AllowOverride All
11 Options +ExecCGI
12 AddHandler cgi-script .cgi .pl .py
13 Order allow,deny
14 allow from all
15 </Directory>
16 ...
</VirtualHost>
```

Kode di atas memerintahkan aplikasi *Apache* untuk meneruskan permintaan pengguna dari alamat dengan format *URL* ke aplikasi yang telah dirilis sehingga aplikasi yang telah dirilis dapat mulai untuk menerima permintaan pengguna.

K. Sprint 11

Tabel 4.18: Sprint 11 Backlog

No	Story	Task	Status
1	<i>Background Task Celery</i>	memindahkan beban kerja proses utama dengan proses yang lainnya dengan <i>Celery</i>	selesai

Aplikasi *Celery* digunakan oleh aplikasi web berbasis *Flask* untuk memindahkan sebagian beban kerja yang besar dari aplikasi web berbasis *flask* yang ada. Dengan adanya pemindahan beban dari aplikasi web berbasis *flask* ke aplikasi *Celery* ini, aplikasi web berbasis *flask* dapat melayani banyak permintaan pengguna yang masuk.

```

1 import sys
2 import os
3
4 sys.path.insert(0, os.path.abspath('..../..'))
5
6 from src.api.app import run
7 from celery import shared_task
8 from src.crawling.crawl import Crawl
9 from src.document_ranking.service import DocumentRankingService
10 from src.page_ranking.page_rank import run_background_service
11 import time
12
13 flask = run()
14 handle = None
15
16 @shared_task(name='workers.document_ranking.run', bind=True)
17 def run_document_ranking(self, algorithm, options):
18
19     start_time = time.time()
20
21     self.update_state(meta={
22         "algorithm": algorithm,
23         "start_time": start_time,
24         "options": options or dict()
25     })
26
27     s = DocumentRankingService(algorithm=algorithm)
28

```

```
29     s.run(options=options)
30
31
32     @shared_task(name='workers.page_ranking.run', bind=True)
33     def run_page_ranking(self, max_iterations, damping_factor):
34
35         def handle_iteration_change(i):
36             self.update_state(meta={
37                 "iterations": i
38             })
39
40         start_time = time.time()
41
42         self.update_state(meta={
43             "max_iterations": max_iterations,
44             "damping_factor": damping_factor,
45             "start_time": start_time,
46         })
47
48         run_background_service({
49             "max_iterations": max_iterations,
50             "damping_factor": damping_factor,
51             "on_iteration_change": handle_iteration_change
52         })
53
54
55     @shared_task(name='workers.crawler.run', bind=True)
56     def run_crawl(self, status, start_urls, max_threads,
57                  bfs_duration_sec, msb_duration_sec, msb_keyword):
58         start_time = time.time()
59
60         c = Crawl(status, start_urls, max_threads, bfs_duration_sec,
61                   msb_duration_sec, msb_keyword)
62
63         self.update_state(meta={
64             "threads": max_threads,
65             "duration": bfs_duration_sec + msb_duration_sec,
66             "start_time": start_time,
67             "end_time": start_time + bfs_duration_sec + msb_duration_sec,
68         })
69         c.run()
```

```

69
70     celery_app = flask.extensions["celery"]
71

```

Kode di atas bertujuan untuk mendefinisikan beberapa *task* yang aplikasi *Celery* akan tangani. Untuk aplikasi *Celery* dapat memulai menjalankan tugasnya, aplikasi *Celery* terlebih dahulu didaftarkan menjadi *daemon process* atau program yang berjalan di latar belakang sistem operasi dengan membuat file yang berisi definisi *daemon process* yang akan didaftarkan.

```

1 [Unit]
2 Description=Search Engine Workers Service
3 After=network.target
4
5 [Service]
6 Type=simple
7 EnvironmentFile=/etc/conf.d/celery
8 WorkingDirectory=/var/www/html/se2/src/celery
9 ExecStart=/opt/rh/rh-python38/root/bin/python -m celery -A
10    workers worker -l INFO --pool=eventlet -n SEARCH_ENGINE_WORKERS
11 ExecStop=/opt/rh/rh-python38/root/bin/python -m celery multi
12    stopwait -A workers worker -l INFO --pool=eventlet -n
13    SEARCH_ENGINE_WORKERS
14 ExecReload=/opt/rh/rh-python38/root/bin/python -m celery multi
15    restart -A workers worker -l INFO --pool=eventlet -n
16    SEARCH_ENGINE_WORKERS
17 Restart=always
18
19 [Install]
20 WantedBy=multi-user.target
21

```

Untuk menjalankan aplikasi *Celery* digunakan perintah sebagai berikut

```
1 systemctl start se-workers
```

Sedangkan untuk memberhentikan aplikasi *Celery* adalah sebagai berikut

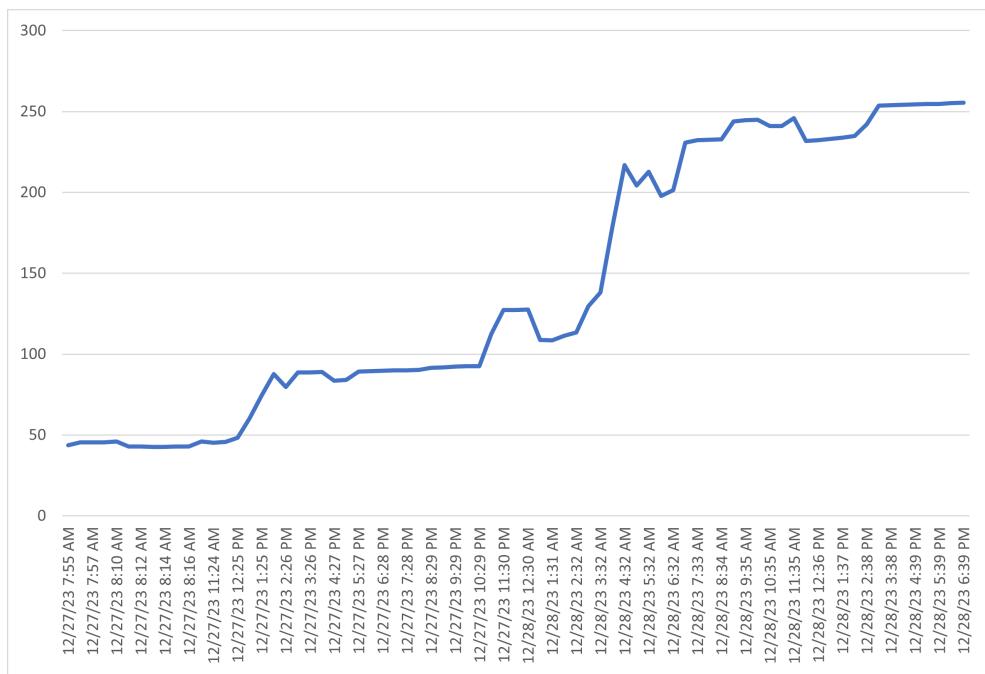
```
1 systemctl stop se-workers
```

L. Sprint 13

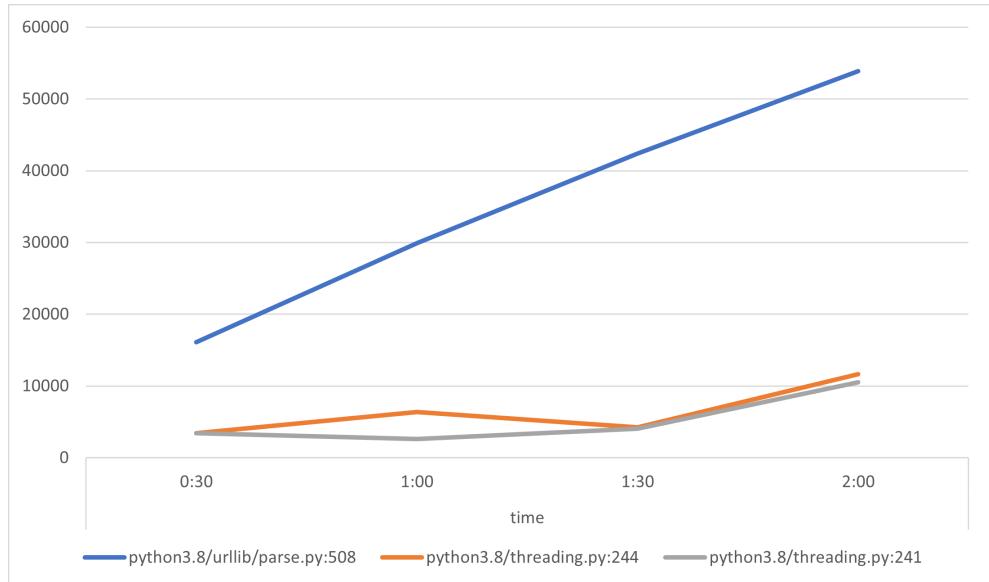
Tabel 4.19: Sprint 13 Backlog

No	Story	Task	Status
1	<i>Memory Profiling</i> atau Pengujian Penggunaan Memori	Memantau penggunaan memori aplikasi	selesai

Dalam *sprint* ini, dilakukan pemujian penggunaan memori aplikasi yang telah dirilis. Pengujian dilakukan dengan menggunakan pustaka *tracemalloc* dan *gc* dari bahasa pemrograman *python*. Pencacatan penggunaan memori aplikasi dilakukan setiap 30 menit dalam waktu lebih dari satu hari di lingkungan *server*. Adapun grafik penggunaan memori aplikasi adalah sebagai berikut.

**Gambar 4.52:** Grafik penggunaan memori aplikasi

Percobaan lain dilakukan dengan waktu yang lebih singkat untuk mengidentifikasi letak kenaikan penggunaan memori oleh aplikasi. Dari percobaan yang dilakukan terdapat kenaikan yang signifikan dari pustaka *urllib* dari bahasa pemrograman *python*. Dapat disimpulkan kenaikan penggunaan memori dari aplikasi disebabkan oleh pustaka dari bahasa pemrograman *python* yang digunakan.



Gambar 4.53: Grafik penggunaan memori aplikasi dari *tracemalloc*

M. Hasil Pengujian *User Acceptance Test*

Tabel 4.20: Hasil Pengujian *User Acceptance Test*

No	<i>Acceptance Requirements</i>	Kesesuaian		Keterangan
		Setuju	Tidak Setuju	
1	Fitur pencarian pengguna sudah sesuai dengan kebutuhan pengguna	✓		
2	Fitur staff sudah sesuai dengan kebutuhan pengguna	✓		
3	Fitur <i>crawling</i> sudah sesuai kebutuhan pengguna	✓		
4	Fitur <i>document ranking</i> sudah sesuai kebutuhan pengguna	✓		
5	Fitur <i>page ranking</i> sudah sesuai kebutuhan pengguna		✓	Butuh Penyesuaian

BAB V

KESIMPULAN DAN SARAN

A. Kesimpulan

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

1. Perancangan aplikasi *admin panel* untuk manajemen sumber daya *search engine* yang telah dibuat pada penelitian sebelumnya yang dirancang menggunakan metode *scrum*.
2. Berdasarkan pengujian *unit testing* yang dilakukan peneliti, aplikasi berjalan dengan baik. Berdasarkan hasil pengujian UAT terdapat feedback untuk suatu fitur yaitu *crawling* pada bagian peta situs. Pada fitur ini terdapat masalah berupa performa yang tidak bagus sehingga disarankan untuk penyederhanaan fitur.

B. Saran

Adapun saran untuk penelitian selanjutnya adalah:

1. Penggunaan memori dari aplikasi yang meningkat secara signifikan dari waktu ke waktu menyebabkan *server* dapat mengalami kehabisan *memory*. Perlu diadakan penulisan kode yang lebih efisien dalam hal penggunaan memori.
2. Pada penelitian dijumpai kasus kasus kecil yang terabaikan yang menyebabkan proses berjalannya aplikasi menjadi terhambat. Hal ini dapat dicegah dengan melakukan pengujian kepada kasus kecil yang mungkin terlewat.

DAFTAR PUSTAKA

- Adam, W. dan Schoger, S. (2018). *Refactoring UI*.
- Adi, P. (2015). Scrum method implementation in a software development project management. *International Journal of Advanced Computer Science and Applications*, 6.
- Ajagbe, S., Oladipupo, M., dan Emmanuel, B. (2020). Crime belt monitoring via data visualization: A case study of folium. 4:35–44.
- Alonso, O. dan Baeza-Yates, R. (2000). A model and software architecture for search results visualization on the www. pages 8 – 16.
- B, D., Salim, S., dan mariam varghese, S. (2016). Performance evaluation of mysql and mongodb databases. *International Journal on Cybernetics and Informatics*, 5:387–394.
- Caldarola, E. G., Picariello, A., Rinaldi, A., dan Sacco, M. (2016). Exploration and visualization of big graphs - the dbpedia case study.
- Fadli, K., Basuki, A., dan Setiawan, E. (2020). Identifikasi malicious host dalam local area network menggunakan teknik graph clustering dan filtering. *Jurnal Teknologi Informasi dan Ilmu Komputer*, 7:591.
- Győrödi, C., Győrödi, R., Pecherle, G., dan Olah, A. (2015). A comparative study: Mongodb vs. mysql. In *2015 13th International Conference on Engineering of Modern Electric Systems (EMES)*, pages 1–6.
- Hu, Y. dan Nöllenburg, M. (2018). *Graph Visualization*, pages 1–9.
- Keat, L. dan Chuah, C. W. (2018). Smart indoor home surveillance monitoring system using raspberry pi. *JOIV : International Journal on Informatics Visualization*, 2.
- Khatulistiwa, L. (2022). Perancangan arsitektur search engine dengan mengintegrasikan web crawler, algoritma page ranking, dan document ranking. *Program Studi Ilmu Komputer Fakultas Matematika Dan Ilmu Pengetahuan Alam Universitas Negeri Jakarta 2022*.

- M. Agus Muhyidin, Muhammad Afif Sulhan, A. S. (2020). Perancangan ui/ux aplikasi my cic layanan informasi akademik mahasiswa menggunakan aplikasi figma.
- M. Shalahuddin, R. A. (2014). *Rekayasa Perangkat Lunak: Terstruktur dan Berorientasi Objek*.
- Mingtao, S. (2010). Software functional testing from the perspective of business practice. *Computer and Information Science*, 3.
- Muller, M. (2007). *Participatory Design*, pages 1061–1081.
- Mungekar, A. (2019). Data storage and management project.
- Perry, W. E. (2006). *Effective Methods for Software Testing: Includes Complete Guidelines, Checklists, and Templates*.
- Satzinger, J., Jackson, R., dan Burd, S. (2011, publisher=). Systems analysis and design in a changing world.
- Sharma, A., Aggarwal, N., Duhan, N., dan Gupta, R. (2010). Web search result optimization by mining the search engine query logs. In *2010 International Conference on Methods and Models in Computer Science (ICM2CS-2010)*, pages 39–45.
- Stabina, R. (2005). Quantitative data graphics: Best practices of designing tables and graphs for use in not-for-profit evaluation reports.