

**PENERAPAN ARSITEKTUR *CLIENT DATA LAYER* PADA  
DASHBOARD SENTIMENT ANALYSIS MENGGUNAKAN  
TANSTACK QUERY DENGAN METODE FOUNTAIN**

**LAPORAN TUGAS AKHIR  
PENGHARGAAN SKEMA JAWA TIMUR DATATHON**



**Oleh:**  
**NASRUL FAHMI ULUMUDDIN**  
**NIM 362258302204**

**PROGRAM STUDI SARJANA TERAPAN  
TEKNOLOGI REKAYASA PERANGKAT LUNAK  
BISNIS DAN INFORMATIKA  
POLITEKNIK NEGERI BANYUWANGI**

**2026**

*– Halaman ini sengaja dikosongkan –*

**PENERAPAN ARSITEKTUR *CLIENT DATA LAYER* PADA  
DASHBOARD SENTIMENT ANALYSIS MENGGUNAKAN  
TANSTACK QUERY DENGAN METODE FOUNTAIN**

**LAPORAN TUGAS AKHIR  
PENGHARGAAN SKEMA JAWA TIMUR DATATHON**



Tugas Akhir ini Dibuat dan Diajukan untuk Memenuhi Salah Satu Syarat Kelulusan  
Program Studi Sarjana Terapan Teknologi Rekayasa Perangkat Lunak  
dan Mencapai Gelar Sarjana Terapan Komputer (S.Tr.Kom)

**Oleh:**  
**NASRUL FAHMI ULUMUDDIN**  
**NIM 362258302204**

**PROGRAM STUDI SARJANA TERAPAN  
TEKNOLOGI REKAYASA PERANGKAT LUNAK  
BISNIS DAN INFORMATIKA  
POLITEKNIK NEGERI BANYUWANGI**

**2026**

*– Halaman ini sengaja dikosongkan –*

## **HALAMAN PERSEMBAHAN**

Dengan segenap rasa syukur yang mendalam, penulis panjatkan puji dan syukur ke hadirat Allah SWT atas segala limpahan rahmat, hidayah, dan karunia-Nya yang tiada terhingga. Atas izin dan kehendak-Nya, tugas akhir ini akhirnya dapat terselesaikan. Pada kesempatan yang penuh makna ini, penulis mempersembahkan karya ini sebagai bentuk penghormatan, cinta, dan rasa terima kasih kepada:

1. Allah SWT, Zat Yang Maha Pengasih dan Maha Penyayang, atas setiap napas kehidupan, kekuatan, kesabaran, serta jalan yang tak pernah henti Engkau berikan. Tanpa ridha-Mu, langkah ini tidak akan pernah sampai sejauh ini.
2. Kepada diri sendiri, terima kasih telah memilih untuk terus berjalan meski lelah, tetap bertahan meski nyaris menyerah. Terima kasih sudah menjadi kuat saat tidak ada yang menguatkan, sudah belajar berdiri di bahu sendiri, melangkah dengan keyakinan, dan tidak pernah berhenti berusaha. Terima kasih telah percaya bahwa setiap perjuangan ini layak untuk dijalani, dan setiap air mata yang jatuh adalah bagian dari proses tumbuh.
3. Ayah dan Ibu tercinta penulis, terima kasih atas semua kasih sayang, doa yang tak pernah putus, dukungan yang begitu besar, dan segala pengorbanan yang tak akan pernah bisa terbalas dengan apa pun. Semua pencapaian ini tidak akan terwujud tanpa restu dan bimbingan dari kalian.
4. Bapak Sepyan Purnama Kristanto, S.Kom., M.Kom dan Ibu Arum Andary Ratri, S.Si., M.Si terima kasih atas segala bimbingan, ilmu, dan kesabaran dalam mendampingi proses penyusunan tugas akhir ini. Setiap arahan dan nasihat dari Ibu sangat berarti dan akan selalu menjadi bekal di masa depan.
5. Bapak Devit Suwardiyanto, S.Si., M.T dan Ibu Ruth Ema Febrita, S.Pd., M.Kom terima kasih atas kritik dan saran yang membangun, serta waktu yang telah Bapak luangkan untuk memberi masukan demi penyempurnaan tugas akhir ini.
6. Ibu Dianni Yusuf, S.Kom., M.Kom yang selalu mendukung, memberikan arahan, dan memberikan hal yang terbaik kepada tim Datathon, serta Mohamad Aji Hermansyah, S.Kom. yang selalu menjadi Kakak mentor di lingkungan kampus.
7. Almamater tercinta dan seluruh dosen TRPL, terima kasih atas segala ilmu, bimbingan, dan kesempatan yang diberikan selama masa perkuliahan. Serta teman-teman seperjuangan kelas E-TRPL Angkatan 2022 yang tidak bisa penulis sebutkan satu persatu, terima kasih atas semangat dan kebersamaan selama proses panjang ini
8. Seluruh mentor-mentor Jagoan Indonesia yang telah memberikan motivasi, bimbingan, serta ilmu yang sangat bermanfaat selama proses magang. Ucapan terima kasih secara khusus penulis sampaikan kepada Bapak Dias Satria, S.E., M.App.Ec., Ph.D., Kakak Puspita Wikanandha, MBA., Kakak Ryo Santoso, S.M., Kakak Handoyo Tejo Wasito, S.M., dan Kakak Diego Irsandy, S.Stat., M.Stat., atas arahan, wawasan, dan pengalaman berharga yang telah dibagikan. Penulis juga mengucapkan terima kasih kepada teman seperjuangan di tempat magang, atas kebersamaan, dukungan, proses pembelajaran dan pengembangan diri.

9. Penulis juga menyampaikan terima kasih kepada Tim Datathon, Ravi Athallah Yuwono, Muhammad Arij Alfarizi, Muhammad Elang Prakoso, Roy Hairul Anam, atas kerja sama, kebersamaan, serta semangat saling mendukung selama proses pembelajaran, diskusi, dan penyelesaian berbagai tantangan, yang turut memberikan pengalaman berharga dan memperkaya perjalanan akademik penulis.

## MOTTO

*“The winning of the game is required you to focus on the card you have, and choosing what to do with them, lay with other player not against them.”*

— Mel Robbins, *The Let Them Theory*

*– Halaman ini sengaja dikosongkan –*

## HALAMAN PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Nasrul Fahmi Ulumuddin  
NIM : 362258302204  
Program Studi : Teknologi Rekayasa Perangkat Lunak  
Jurusan : Bisnis dan Informatika

Menyatakan bahwa Tugas Akhir yang berjudul "Penerapan Arsitektur *Client Data Layer* pada Dashboard *Sentiment Analysis* Menggunakan TanStack Query dengan Metode Fountain" adalah benar-benar hasil karya sendiri, kecuali jika disebutkan sumbernya dan belum pernah diajukan di institusi manapun, serta bukan karya jiplakan/plagiat. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa adanya tekanan dan paksaan dari pihak manapun serta bersedia mendapat sanksi akademik jika ternyata di kemudian hari pernyataan ini tidak benar.

Banyuwangi, 10 Februari 2026

Yang membuat pernyataan,

Nasrul Fahmi Ulumuddin

NIM. 362258302204

*– Halaman ini sengaja dikosongkan –*

**PENERAPAN ARSITEKTUR *CLIENT DATA LAYER* PADA  
DASHBOARD SENTIMENT ANALYSIS MENGGUNAKAN  
TANSTACK QUERY DENGAN METODE FOUNTAIN**

**Tugas Akhir Ini Disusun Untuk Memenuhi Salah Satu Syarat Memperoleh  
Gelar Sarjana Terapan Komputer (S.Tr.Kom)  
Politeknik Negeri Banyuwangi**

**oleh:**

**NASRUL FAHMI ULUMUDDIN**

**NIM 362258302204**

**Tanggal Ujian : 29 Januari 2026**

**Menyetujui,**

**Pembimbing 1 : Sepyan Purnama Kristanto, S.Kom., M.Kom (.....)**

**Pembimbing 2 : Arum Andary Ratri, S.Si., M.Si. (.....)**

**Pengaji 1 : Devit Suwardiyanto,S.Si., M.T. (.....)**

**Pengaji 2 : Ruth Ema Febrita, S.Pd., M.Kom. (.....)**

**Mengesahkan,  
Ketua Jurusan  
Bisnis dan Informatika**

**Mengetahui,  
Koordinator Program Studi  
Teknologi Rekayasa Perangkat Lunak**

**I Wayan Suardinata, S.Kom., M.T.  
NIP. 198010222015041001**

**Lutfi Hakim, S.Pd., M.T.  
NIP. 199203302019031012**

*– Halaman ini sengaja dikosongkan –*

# **Penerapan Arsitektur *Client Data Layer* pada Dashboard *Sentiment Analysis* Menggunakan TanStack Query dengan Metode Fountain**

Nama : Nasrul Fahmi Ulumuddin  
NIM : 362258302204  
Pembimbing : 1 Sepyan Purnama Kristanto, S.Kom., M.Kom  
              2 Arum Andary Ratri, S.Si., M.Si.

## **ABSTRAK**

Di era digital, pelaku UMKM semakin bergantung pada media sosial seperti Instagram untuk memahami persepsi publik melalui data komentar pengguna yang bersifat tidak terstruktur dan dinamis. Data tersebut umumnya diolah menggunakan *Sentiment Analysis* berbasis *Natural Language Processing* (NLP) dan disajikan dalam bentuk dashboard analitik berbasis web untuk mendukung pemantauan dan pengambilan keputusan. Namun, pengembangan dashboard menghadapi tantangan pada pengelolaan alur data *frontend*, seperti permintaan API berulang dan inkonsistensi data antar-komponen. Oleh karena itu, penelitian ini bertujuan menerapkan arsitektur *Client Data Layer* menggunakan pustaka TanStack Query guna menciptakan pengelolaan data *frontend* yang lebih terstruktur serta menganalisis perilaku sistem terkait konsistensi data dan efisiensi pengambilan data melalui mekanisme *caching*.

Penelitian ini dilaksanakan menggunakan metode Fountain yang bersifat iteratif dan fleksibel, memungkinkan tahapan analisis, perancangan, implementasi, dan pengujian dilakukan secara tumpang tindih. Sistem dikembangkan sebagai aplikasi *frontend* berbasis React dengan TanStack Query sebagai solusi *server state management* untuk mengelola proses pengambilan, penyimpanan sementara, dan sinkronisasi data dari REST API backend. Pengujian sistem dilakukan menggunakan pendekatan *black-box testing* berbasis skenario untuk mengevaluasi mekanisme pengelolaan data *frontend*, meliputi konsistensi data antar-komponen, efektivitas mekanisme *caching*, serta pengendalian pemanggilan API pada berbagai skenario penggunaan.

Hasil penelitian menunjukkan bahwa penerapan arsitektur *Client Data Layer* menggunakan TanStack Query mampu menjaga konsistensi informasi di seluruh bagian dashboard tanpa memicu permintaan API tambahan saat pengguna melakukan navigasi antar-halaman. Pengujian terhadap 37 skenario fungsional menunjukkan tingkat keberhasilan yang tinggi, meskipun ditemukan satu kegagalan minor pada representasi state awal pemuatan data. Secara keseluruhan, penelitian ini menyimpulkan bahwa penerapan TanStack Query efektif meningkatkan efisiensi komunikasi antara *frontend* dan backend serta mendukung pengembangan dashboard analitik yang lebih terstruktur, stabil, dan skalabel.

**Kata kunci:** *Client Data Layer*, TanStack Query, Dashboard Analitik, *Frontend Data-Driven*, Manajemen *Server State*

*– Halaman ini sengaja dikosongkan –*

# **Implementation of Client Data Layer Architecture using TanStack Query in a Sentiment Analysis Dashboard with the Fountain Method**

Name : Nasrul Fahmi Ulumuddin  
NIM : 362258302204  
Advisors : 1 Sepyan Purnama Kristanto, S.Kom., M.Kom  
              2 Arum Andary Ratri, S.Si., M.Si.

## **ABSTRACT**

*In the digital era, Micro, Small, and Medium Enterprises (MSMEs) increasingly rely on social media platforms such as Instagram to understand public perception through unstructured and dynamic user comment data. These data are commonly processed using Natural Language Processing (NLP)-based sentiment analysis and presented through web-based analytical dashboards to support monitoring and decision-making. However, the development of such dashboards introduces challenges in frontend data flow management, including repeated API requests and data inconsistency across components. Therefore, this study aims to implement a Client Data Layer architecture using the TanStack Query library and to analyze frontend system behavior in terms of data consistency and data retrieval efficiency through caching mechanisms.*

*This research was conducted using the Fountain method, which supports iterative and overlapping phases of analysis, design, implementation, and testing. The system was developed as a React-based frontend application, with TanStack Query employed as a server state management solution to handle data fetching, temporary storage, and synchronization from a backend REST API. System evaluation was carried out using a scenario-based black-box testing approach to examine frontend data management behavior, including data consistency between components, caching effectiveness, and API request control under various usage scenarios.*

*The results indicate that the implementation of the Client Data Layer architecture using TanStack Query successfully maintains information consistency across all dashboard components without triggering redundant API requests during page navigation. Testing across 37 functional scenarios demonstrated a high success rate, with only one minor issue identified in the initial data loading state representation. Overall, this study concludes that the use of TanStack Query effectively improves frontend–backend communication efficiency and supports the development of more structured, stable, and scalable data-driven analytical dashboards.*

**Key words:** Client Data Layer, TanStack Query, Analytical Dashboard, Data-Driven Frontend, Server State Management

*– Halaman ini sengaja dikosongkan –*

## KATA PENGANTAR

Puji syukur kehadirat Allah SWT atas berkat rahmat dan karunia-Nya, Tugas Akhir dalam rangka untuk memenuhi sebagian persyaratan untuk mendapatkan gelar Sarjana Terapan.

Tugas Akhir ini dapat diselesaikan tidak lepas dari bantuan dan kerjasama dengan pihak lain. Berkenaan dengan hal tersebut, penulis menyampaikan ucapan terima kasih kepada yang terhormat:

1. Bapak M. Shofiu Amin, S.T., M.T. selaku Direktur Politeknik Negeri Banyuwangi.
2. Bapak I Wayan Suardinata, S.Kom., M.T. selaku Ketua Bisnis dan Informatika Politeknik Negeri Banyuwangi.
3. Bapak Lutfi Hakim, S.Pd., M.T. selaku Koordinator Program Studi Teknologi Rekayasa Perangkat Lunak Politeknik Negeri Banyuwangi.
4. Bapak Sepyan Purnama Kristanto, S.Kom., M.Kom selaku Dosen Pembimbing 1 yang telah banyak memberikan semangat, dorongan, dan bimbingan selama penyusunan Tugas Akhir ini.
5. Ibu Arum Andary Ratri, S.Si., M.Si. selaku Dosen Pembimbing 2 yang telah banyak memberikan semangat, dorongan, dan bimbingan selama penyusunan Tugas Akhir ini.
6. Bapak Devit Suwardiyanto,S.Si., M.T. dan Ibu Ruth Ema Febrita, S.Pd., M.Kom. selaku Tim Penguji yang sudah memberikan koreksi perbaikan secara komprehensif terhadap Tugas Akhir ini.
7. Bapak/Ibu dosen dan staff Program Studi Teknologi Rekayasa Perangkat Lunak yang telah memberikan ilmu dan bantuan selama masa perkuliahan.
8. Orang tua penulis yang selalu mendoakan dan memberikan motivasi sehingga penulis dapat menyelesaikan laporan tugas akhir.
9. Keluarga penulis tercinta yang selalu memberikan doa, dukungan, dan semangat.
10. Teman-teman penulis yang telah memberikan bantuan, dukungan moral, dan motivasi selama penyusunan laporan ini.

Penulis menyadari bahwa Laporan Tugas Akhir ini masih jauh dari kata sempurna, oleh karena itu, penulis mengharapkan kritik dan saran yang bersifat membangun untuk menyempurnakan laporan ini menjadi lebih baik. Semoga laporan ini dapat memberikan manfaat bagi pembaca dan pihak-pihak yang berkepentingan.

Banyuwangi, 10 Februari 2026

Penulis

*– Halaman ini sengaja dikosongkan –*

## DAFTAR ISI

<b>HALAMAN SAMPUL.....</b>	<b>i</b>
<b>HALAMAN PERSEMBAHAN.....</b>	<b>iii</b>
<b>HALAMAN PENGESAHAN .....</b>	<b>ix</b>
<b>ABSTRAK .....</b>	<b>xi</b>
<b>ABSTRACT.....</b>	<b>xiii</b>
<b>KATA PENGANTAR .....</b>	<b>xv</b>
<b>DAFTAR ISI .....</b>	<b>xvii</b>
<b>DAFTAR SINGKATAN .....</b>	<b>xix</b>
<b>DAFTAR GAMBAR .....</b>	<b>xxi</b>
<b>DAFTAR TABEL .....</b>	<b>xxiii</b>
<b>BAB 1 PENDAHULUAN .....</b>	<b>1</b>
1.1 Latar Belakang .....	1
1.2 Rumusan Masalah .....	2
1.3 Tujuan .....	3
1.4 Manfaat.....	3
1.5 Batasan Masalah .....	3
<b>BAB 2 TINJAUAN PUSTAKA .....</b>	<b>5</b>
2.1 Landasan Teori .....	5
2.1.1 UMKM dan Digitalisasi .....	5
2.1.2 Media Sosial sebagai Sumber Data .....	6
2.1.3 Analisis Sentimen pada Media Sosial .....	6
2.1.4 Dashboard Analitik dan Visualisasi Data.....	7
2.1.5 <i>Blackbox Testing</i> .....	9
2.1.6 Arsitektur <i>Client Data Layer</i> .....	9
2.1.7 <i>REST API</i> .....	15
2.1.8 React .....	16
2.1.9 TanStack Query (React Query) .....	17
2.1.10 Metode Fountain.....	18
2.2 Penelitian Terkait .....	21
<b>BAB 3 METODOLOGI PENELITIAN.....</b>	<b>25</b>
3.1 Waktu dan Jadwal penelitian .....	25
3.1.1 Waktu Pelaksanaan Penelitian .....	25
3.1.2 Jadwal Kegiatan Penelitian.....	25
3.2 Metode Fountain .....	27

3.2.1	<i>Analysis</i> .....	28
3.2.2	<i>Requirement Specification</i> .....	29
3.2.3	<i>Design</i> .....	30
3.2.4	<i>Coding (Implementation)</i> .....	42
3.2.5	<i>Testing</i> .....	43
<b>BAB 4</b>	<b>HASIL DAN PEMBAHASAN</b> .....	<b>57</b>
4.1	Hasil Jawa Timur Datathon 2025 .....	57
4.1.1	Capaian Jawa Timur Datathon 2025 Skema UMKM .....	57
4.2	Hasil .....	60
4.2.1	<i>Analysis</i> .....	60
4.2.2	<i>Requirement Specification</i> .....	60
4.2.3	<i>Design</i> .....	60
4.2.4	<i>Coding/Implementation</i> .....	63
4.2.5	<i>Testing</i> .....	70
4.3	Pembahasan .....	80
4.3.1	Peran Arsitektur <i>Client Data Layer</i> dalam Pengelolaan Data <i>Frontend</i> ...	80
4.3.2	Analisis Implementasi TanStack Query terhadap Server State .....	81
4.3.3	Implikasi Metode Fountain terhadap Iterasi Pengembangan Dashboard ..	89
4.3.4	Analisis Hasil dan Temuan Pengujian Sistem .....	90
4.3.5	Perbandingan dengan Penelitian Terdahulu dan Landasan Teori.....	90
4.3.6	Dampak terhadap Skalabilitas dan Maintainability Sistem .....	91
4.3.7	Keterbatasan Penerapan TanStack Query dalam Penelitian .....	92
<b>BAB 5</b>	<b>KESIMPULAN DAN SARAN</b> .....	<b>93</b>
5.1	Kesimpulan .....	93
5.2	Saran.....	93
<b>DAFTAR PUSTAKA</b>	.....	<b>95</b>
<b>LAMPIRAN</b>	.....	<b>97</b>
Lampiran 1.	<i>Source Code</i> .....	97
Lampiran 2.	Dokumentasi Bimbingan .....	101
Lampiran 3.	Dokumentasi Penyerahan Hadiah .....	101
Lampiran 4.	Dokumentasi dengan Dosen Pembimbing .....	102
Lampiran 5.	Sertifikat .....	102

## **DAFTAR SINGKATAN**

UMKM	: <i>Usaha Mikro, Kecil, dan Menengah</i>
NLP	: <i>Natural Language Processing</i>
API	: <i>Application Programming Interface</i>
UI	: <i>User Interface</i>
HTTP	: <i>Hypertext Transfer Protocol</i>
REST	: <i>Representational State Transfer</i>
JSON	: <i>JavaScript Object Notation</i>

*– Halaman ini sengaja dikosongkan –*

## DAFTAR GAMBAR

<b>Gambar 2.1</b>	Arsitektur <i>Client Data Layer</i> .....	10
<b>Gambar 2.2</b>	Fountain SDLC Model .....	19
<b>Gambar 3.1</b>	Use Case Diagram .....	30
<b>Gambar 3.2</b>	Entity Relationship Diagram .....	32
<b>Gambar 3.3</b>	Alur Sistem Keseluruhan .....	33
<b>Gambar 3.4</b>	Diagram Arsitektur <i>Frontend</i> .....	35
<b>Gambar 3.5</b>	Cara Kerja TanStack Query .....	37
<b>Gambar 3.6</b>	Wireframe Landing Page .....	38
<b>Gambar 3.7</b>	Wireframe Halaman Login .....	39
<b>Gambar 3.8</b>	Wireframe Halaman Register .....	39
<b>Gambar 3.9</b>	Wireframe Halaman Dashboard .....	40
<b>Gambar 3.10</b>	Wireframe Halaman Sentiment .....	40
<b>Gambar 3.11</b>	Wireframe Halaman Scraper .....	41
<b>Gambar 3.12</b>	Wireframe Halaman Recomendation .....	41
<b>Gambar 3.13</b>	Wireframe Halaman Chatbot .....	42
<b>Gambar 4.1</b>	Pengumuman Juara 3 Jawa Timur Datathon 2025 .....	59
<b>Gambar 4.2</b>	Implementasi Antarmuka Landing Page .....	64
<b>Gambar 4.3</b>	Implementasi Antarmuka Halaman Login .....	65
<b>Gambar 4.4</b>	Implementasi Antarmuka Halaman Register .....	65
<b>Gambar 4.5</b>	Implementasi Antarmuka Halaman Dashboard .....	66
<b>Gambar 4.6</b>	Implementasi Antarmuka Halaman Scraper .....	67
<b>Gambar 4.7</b>	Implementasi Antarmuka Halaman Sentiment .....	68
<b>Gambar 4.8</b>	Implementasi Antarmuka Halaman Rekomendasi Konten .....	69
<b>Gambar 4.9</b>	Desain Komponen Chatbot .....	70
<b>Gambar 4.10</b>	Struktur Folder Per Fitur .....	81
<b>Gambar 4.11</b>	Konfigurasi Query Client .....	82
<b>Gambar 4.12</b>	Konfigurasi Query Keys .....	82
<b>Gambar 4.13</b>	Konfigurasi Repository .....	83
<b>Gambar 4.14</b>	Konfigurasi Scraper .....	83
<b>Gambar 4.15</b>	Konfigurasi Repository .....	84
<b>Gambar 4.16</b>	Konfigurasi Scraper .....	84
<b>Gambar 4.17</b>	Diagram Alur Pengelolaan Query menggunakan TanStack Query .....	86
<b>Gambar 4.18</b>	Diagram Alur Pengelolaan Mutation menggunakan TanStack Query .....	87

*– Halaman ini sengaja dikosongkan –*

## DAFTAR TABEL

<b>Tabel 2.1</b>	Tabel Perbandingan Penelitian Terkait .....	22
<b>Tabel 3.1</b>	Jadwal Pelaksanaan Penelitian .....	25
<b>Tabel 3.2</b>	Skenario Pengujian Landing Page .....	44
<b>Tabel 3.3</b>	Skenario Pengujian Halaman Login .....	44
<b>Tabel 3.4</b>	Skenario Pengujian Halaman Register .....	46
<b>Tabel 3.5</b>	Skenario Pengujian Halaman Dashboard .....	47
<b>Tabel 3.6</b>	Skenario Pengujian Halaman Dashboard Sentiment .....	49
<b>Tabel 3.7</b>	Skenario Pengujian Halaman Dashboard Rekomendasi Konten .....	50
<b>Tabel 3.8</b>	Skenario Pengujian Halaman Data Scraper .....	51
<b>Tabel 3.9</b>	Skenario Pengujian Chatbot .....	53
<b>Tabel 3.10</b>	Skenario Pengujian Logout .....	54
<b>Tabel 4.1</b>	Kriteria Penilaian Pengumpulan Proposal Ide Analisis Data dan Video .....	57
<b>Tabel 4.2</b>	Kriteria Penilaian Babak Final Jawa Timur Datathon 2025 .....	58
<b>Tabel 4.3</b>	Hasil Pengujian Landing Page .....	71
<b>Tabel 4.4</b>	Hasil Pengujian Fitur Login .....	71
<b>Tabel 4.5</b>	Hasil Pengujian Fitur Register .....	72
<b>Tabel 4.6</b>	Hasil Pengujian Fitur Dashboard .....	73
<b>Tabel 4.7</b>	Hasil Pengujian Halaman Dashboard Sentiment .....	75
<b>Tabel 4.8</b>	Hasil Pengujian Halaman Dashboard Rekomendasi Konten .....	76
<b>Tabel 4.9</b>	Hasil Pengujian Halaman Data Scraper .....	77
<b>Tabel 4.10</b>	Hasil Pengujian Fitur Chatbot .....	78
<b>Tabel 4.11</b>	Hasil Pengujian Fitur Logout .....	79

*– Halaman ini sengaja dikosongkan –*

# BAB 1

## PENDAHULUAN

### 1.1 Latar Belakang

Di era digital saat ini, perilaku konsumen telah berubah secara signifikan. Media sosial seperti Instagram, tidak hanya menjadi saluran pemasaran, tetapi juga menjadi ruang interaksi antara konsumen dan pelaku usaha. Studi oleh (Trulline, 2021) menunjukkan bahwa UMKM semakin bergantung pada media sosial untuk mempromosikan produk, membangun reputasi, dan memahami minat pasar. Komentar konsumen pada postingan media sosial mencerminkan persepsi publik terhadap produk atau layanan UMKM, sehingga dapat dimanfaatkan sebagai indikator kepuasan dan reputasi merek.

Namun, data komentar media sosial bersifat tidak terstruktur, jumlahnya besar, dan berubah secara dinamis, sehingga sulit dianalisis tanpa bantuan teknologi. Sejalan dengan (Joseph, 2024), *Sentiment Analysis* pada media sosial merupakan bidang penelitian yang berkembang pesat dan melibatkan penggunaan *Natural Language Processing* (NLP), *text analysis*, dan *computational linguistics* untuk mengidentifikasi serta mengekstraksi informasi subjektif dari data teks. Pendekatan NLP tersebut memungkinkan komentar tidak terstruktur diproses menjadi kategori sentimen yang dapat diinterpretasikan, baik positif, negatif, maupun netral. Dengan demikian, hasil analisis sentimen dari proses NLP dapat disajikan dalam bentuk dashboard interaktif agar UMKM dapat memahami tren sentimen, isu yang sering muncul, serta persepsi pelanggan secara lebih cepat dan intuitif.

Dalam pengembangan dashboard analitik berbasis web, tantangan tidak hanya terletak pada penyajian visualisasi data, tetapi juga pada pengelolaan alur data di sisi *frontend*. Dashboard analitik merupakan aplikasi yang bersifat data-driven, di mana berbagai komponen antarmuka seperti grafik, tabel, dan indikator bergantung pada data yang sama yang bersumber dari dataset historis hasil scraping media sosial dan diproses atas permintaan pengguna. Jika pengambilan data dilakukan secara langsung pada setiap komponen tanpa arsitektur pengelolaan data yang terstruktur, maka dapat muncul berbagai permasalahan, seperti permintaan API berulang dan inkonsistensi data antar-komponen. Permasalahan tersebut tidak hanya berdampak pada aspek teknis pengembangan aplikasi, tetapi juga berimplikasi langsung terhadap pemanfaatan dashboard oleh pelaku UMKM, karena informasi yang ditampilkan berpotensi tidak akurat atau tidak konsisten antar-halaman, sehingga dapat menyebabkan interpretasi yang keliru terhadap tren sentimen dan kondisi pasar, menurunkan kepercayaan terhadap sistem, serta menurunkan

efektivitas pemanfaatan data sebagai dasar pengambilan keputusan yang berpotensi berdampak pada keberlanjutan usaha UMKM.

Pendekatan arsitektur *Client Data Layer* hadir sebagai solusi untuk mengelola data yang bersumber dari server secara terpusat di sisi klien. Dengan adanya *Client Data Layer*, proses pengambilan, *caching*, dan sinkronisasi data dapat dilakukan secara terstruktur, sehingga komponen antarmuka tidak perlu berinteraksi langsung dengan API. Salah satu pustaka yang mendukung pendekatan ini adalah TanStack Query, yang dirancang untuk mengelola server state secara efisien melalui mekanisme caching, deduplikasi permintaan, serta sinkronisasi data antar-komponen.

Sejumlah penelitian terdahulu menunjukkan bahwa TanStack Query memiliki keunggulan dalam pengelolaan data asinkron pada aplikasi *frontend* dibandingkan pendekatan pengelolaan data konvensional, khususnya pada aplikasi yang bersifat data-driven dan menampilkan data dalam jumlah besar (Luz, 2025). Penelitian yang membandingkan TanStack Query dengan pustaka state management lain juga melaporkan adanya peningkatan efisiensi pengelolaan server state dan pengurangan permintaan data yang tidak diperlukan (Micheal, 2025). Namun demikian, sebagian besar penelitian tersebut masih berfokus pada aspek teknis pustaka atau perbandingan antar-library, dan belum secara spesifik mengkaji penerapan TanStack Query sebagai *Client Data Layer* pada studi kasus dashboard analitik.

Berdasarkan permasalahan tersebut, penelitian ini muncul dari hipotesis bahwa penerapan arsitektur *Client Data Layer* menggunakan TanStack Query pada aplikasi dashboard analitik mampu menghasilkan pengelolaan data *frontend* yang lebih terstruktur dan konsisten dibandingkan pendekatan pengelolaan data konvensional, yaitu pengambilan data API secara langsung pada komponen antarmuka tanpa mekanisme pengelolaan data terpusat. Perilaku tersebut diasumsikan tercermin melalui pemanfaatan mekanisme *caching*, pengendalian permintaan data dari API, serta konsistensi data yang ditampilkan pada berbagai komponen dashboard. Untuk menguji hipotesis tersebut, TanStack Query diterapkan pada studi kasus pengembangan Dashboard Analisis Sentimen UMKM dengan metode *Fountain*, dan hasilnya dievaluasi melalui observasi perilaku sistem menggunakan pendekatan *blackbox testing*.

## 1.2 Rumusan Masalah

1. Bagaimana menerapkan arsitektur *Client Data Layer* menggunakan TanStack Query pada pengembangan Dashboard Analisis Sentimen UMKM?
2. Bagaimana mekanisme pengelolaan data *frontend* yang dihasilkan setelah penerapan TanStack Query ditinjau mekanisme *caching*, dan konsistensi data antar-komponen dashboard?

### **1.3 Tujuan**

1. Menerapkan arsitektur *Client Data Layer* menggunakan TanStack Query pada studi kasus Dashboard Analisis Sentimen UMKM.
2. Menganalisis mekanisme pengelolaan data *frontend* setelah penerapan TanStack Query, khususnya terkait konsistensi data, caching, dan permintaan API.

### **1.4 Manfaat**

#### **Manfaat Teoritis**

1. Penelitian ini diharapkan dapat menambah literatur mengenai penerapan TanStack Query dalam arsitektur data layer pada studi kasus aplikasi dashboard.

#### **Manfaat Praktis**

##### **1. bagi UMKM**

Penelitian ini menghasilkan sebuah dashboard analisis sentimen yang dapat membantu UMKM memahami persepsi konsumen berdasarkan data media sosial secara lebih terstruktur dan mudah dipahami.

##### **2. Manfaat bagi Pengembang**

Penelitian ini memberikan studi kasus penerapan arsitektur *Client Data Layer* menggunakan TanStack Query yang dapat dijadikan acuan dalam merancang pengelolaan data *frontend* pada aplikasi data-driven.

##### **3. Manfaat bagi Akademisi**

Penelitian ini dapat dijadikan referensi bagi penelitian sejenis yang membahas arsitektur *frontend*, pengelolaan server state, dan pengembangan dashboard analitik.

### **1.5 Batasan Masalah**

Batasan proyek ditetapkan agar penelitian tetap terfokus pada tujuan yang telah dirumuskan. Adapun batasan proyek dalam Tugas Akhir ini adalah sebagai berikut:

1. Studi kasus penelitian difokuskan pada Dashboard Analisis Sentimen UMKM dengan ruang lingkup penelitian terbatas pada sisi *frontend* aplikasi.
2. Penelitian hanya membahas penerapan arsitektur *Client Data Layer* menggunakan TanStack Query pada aplikasi *frontend* berbasis React.
3. Proses analisis sentimen, termasuk pengumpulan data media sosial dan pemrosesan teks, tidak dibahas dalam penelitian ini dan sepenuhnya dilakukan pada sisi backend.
4. Penelitian ini tidak melakukan perbandingan implementasi TanStack Query dengan pustaka atau framework state management lain
5. Evaluasi sistem dilakukan menggunakan pendekatan *blackbox testing*, tanpa melakukan pengujian performa numerik atau *benchmarking* mendalam.

6. Penelitian ini tidak membahas aspek optimasi backend, keamanan aplikasi, pengujian beban, maupun pengujian kegunaan secara mendalam.

## **BAB 2**

### **TINJAUAN PUSTAKA**

#### **2.1 Landasan Teori**

##### **2.1.1 UMKM dan Digitalisasi**

Usaha Mikro, Kecil, dan Menengah (UMKM) merupakan sektor usaha produktif yang memiliki peran penting dalam perekonomian, khususnya dalam menciptakan lapangan kerja dan mendorong pertumbuhan ekonomi lokal. UMKM umumnya memiliki karakteristik berupa skala usaha yang relatif kecil, keterbatasan modal, serta pengelolaan usaha yang masih sederhana. Dalam beberapa tahun terakhir, perkembangan teknologi digital telah mendorong UMKM untuk beradaptasi dengan perubahan lingkungan bisnis agar tetap mampu bersaing di tengah dinamika pasar yang semakin cepat.

Meskipun digitalisasi menawarkan berbagai peluang, UMKM masih menghadapi sejumlah tantangan dalam proses adopsinya. Tantangan tersebut meliputi rendahnya literasi digital, keterbatasan pemahaman dalam pemanfaatan teknologi informasi, serta kurangnya kemampuan dalam mengelola dan menganalisis data bisnis. Selain itu, salah satu kendala utama yang dihadapi UMKM adalah keterbatasan akses terhadap sumber daya yang dibutuhkan untuk mengembangkan usaha, seperti modal, informasi, dan teknologi (Alviani et al., 2025).

Permasalahan digitalisasi UMKM juga diperkuat oleh faktor demografis dan geografis. Menurut (Sofyan and Agusman, 2025), rendahnya literasi digital, khususnya pada pelaku UMKM usia lanjut dan yang berada di wilayah terpencil, menjadi hambatan dalam proses transformasi digital. Selain itu, tidak semua UMKM memiliki perangkat pendukung dan akses jaringan internet yang memadai, sehingga proses digitalisasi belum dapat diterapkan secara merata.

Dalam konteks pemasaran, pemanfaatan teknologi digital, khususnya media sosial, telah menjadi salah satu sarana utama bagi UMKM untuk mempromosikan produk dan menjangkau konsumen secara lebih luas. Aktivitas pemasaran digital tersebut menghasilkan data interaksi konsumen dalam jumlah besar yang berpotensi memberikan informasi berharga mengenai persepsi dan preferensi pasar. Oleh karena itu, UMKM membutuhkan sistem informasi yang mampu mengolah dan menyajikan data tersebut secara terstruktur dan informatif. Keberadaan sistem informasi berbasis dashboard analitik menjadi penting untuk mendukung pengambilan keputusan berbasis data serta meningkatkan efektivitas strategi pemasaran UMKM di era digital (Trulline, 2021).

### **2.1.2 Media Sosial sebagai Sumber Data**

Media sosial telah berkembang tidak hanya sebagai sarana komunikasi dan pemasaran, tetapi juga sebagai sumber data yang mencerminkan opini, persepsi, dan perilaku konsumen. Melalui media sosial, konsumen dapat mengekspresikan pengalaman serta keterlibatan mereka terhadap suatu merek melalui berbagai bentuk interaksi, seperti komentar, unggahan, tanda suka, dan aktivitas berbagi konten. Interaksi tersebut mencerminkan keterlibatan konsumen dan menghasilkan data yang bernilai untuk dianalisis lebih lanjut (Mardhatilah et al., 2024). Data yang dihasilkan dari media sosial umumnya bersifat tidak terstruktur dan terus bertambah secara dinamis, sehingga memerlukan sistem yang mampu mengelola dan menyajikan informasi tersebut secara terstruktur agar dapat dimanfaatkan secara optimal.

Data yang dihasilkan dari media sosial memiliki karakteristik bersifat tidak terstruktur, berjumlah besar, dan terus bertambah secara dinamis. Komentar dan ulasan konsumen umumnya berbentuk teks bebas yang mengandung opini subjektif, emosi, serta penilaian terhadap suatu produk atau layanan. Kondisi ini menyebabkan data media sosial sulit dianalisis secara manual. Oleh karena itu, diperlukan pendekatan sistematis untuk mengolah dan mengekstraksi informasi penting dari data tersebut agar dapat dimanfaatkan secara optimal.

Dalam konteks UMKM, data media sosial berpotensi memberikan wawasan penting terkait preferensi konsumen, tingkat kepuasan pelanggan, serta isu-isu yang sering muncul dalam interaksi publik. Informasi ini dapat dimanfaatkan sebagai dasar evaluasi strategi pemasaran dan pengambilan keputusan bisnis. Namun, tanpa dukungan sistem yang mampu mengelola dan menyajikan data secara terstruktur, potensi data media sosial tersebut sulit dimanfaatkan secara efektif.

Oleh karena itu, media sosial diposisikan sebagai salah satu sumber data utama dalam pengembangan sistem analitik bagi UMKM. Data yang diperoleh dari media sosial selanjutnya dapat diolah dan disajikan dalam bentuk informasi yang lebih ringkas dan mudah dipahami melalui dashboard analitik. Pendekatan ini memungkinkan UMKM untuk memantau persepsi konsumen dan dapat digunakan sebagai alat pengambilan keputusan.

### **2.1.3 Analisis Sentimen pada Media Sosial**

Analisis sentimen merupakan pendekatan analitik yang digunakan untuk mengidentifikasi dan mengklasifikasikan opini atau sikap pengguna terhadap suatu objek, seperti produk, layanan, atau merek, berdasarkan data teks. Dalam konteks media sosial, analisis sentimen memanfaatkan komentar, ulasan, dan berbagai bentuk interaksi pengguna untuk menentukan kecenderungan sentimen yang umumnya dikategorikan ke dalam sentimen positif, negatif, atau netral (Suryani

and Komputer, 2024).

Seiring dengan meningkatnya aktivitas pengguna di media sosial, volume data opini yang dihasilkan semakin besar, bersifat dinamis, dan umumnya tidak terstruktur. Data tersebut mengandung unsur subjektivitas, emosi, serta bahasa informal, sehingga sulit dianalisis secara manual. Oleh karena itu, analisis sentimen digunakan untuk menyederhanakan data teks yang kompleks menjadi informasi yang lebih terstruktur agar dapat dimanfaatkan dalam memahami kecenderungan opini publik dan perilaku konsumen.

Penelitian oleh (Fajarini et al., 2025) menunjukkan bahwa analisis sentimen berbasis data media sosial mampu memberikan wawasan mengenai pola opini publik, preferensi konsumen, serta perubahan tren yang relevan dalam konteks bisnis. Pendekatan ini memungkinkan pemantauan opini dan dinilai lebih efisien dibandingkan metode konvensional, seperti survei manual atau riset pasar tradisional.

Selain itu, penelitian lain yang dilakukan oleh (Maharani, 2024) menegaskan bahwa pengolahan data media sosial melalui teknik data mining berperan penting dalam mengidentifikasi pola dan tren penggunaan data yang mencerminkan perilaku serta preferensi pengguna. Dalam konteks tersebut, analisis sentimen menjadi salah satu komponen yang berkontribusi dalam menggali makna dari data teks yang dihasilkan pengguna media sosial, sehingga wawasan yang diperoleh dapat dimanfaatkan untuk mendukung proses analisis dan pengambilan keputusan berbasis data.

Meskipun demikian, analisis sentimen juga memiliki keterbatasan, terutama terkait kualitas data yang bervariasi dan adanya noise dalam data media sosial yang dapat memengaruhi hasil interpretasi. Oleh karena itu, diperlukan pemahaman konteks dan pendekatan analisis yang tepat agar kesimpulan yang dihasilkan tidak bersifat keliru atau bias (Suryani and Komputer, 2024).

Dalam penelitian ini, analisis sentimen diposisikan sebagai proses pengolahan data pada sisi backend. Fokus penelitian tidak terletak pada metode atau algoritma analisis sentimen yang digunakan, melainkan pada pemanfaatan hasil analisis sentimen sebagai sumber data yang dikelola dan disajikan melalui dashboard analitik di sisi *frontend*.

#### **2.1.4 Dashboard Analitik dan Visualisasi Data**

Dashboard analitik merupakan sistem informasi yang dirancang untuk menyajikan data dalam bentuk visual yang ringkas, terintegrasi, dan mudah dipahami oleh pengguna. Dashboard ini umumnya memanfaatkan berbagai komponen visualisasi, seperti grafik, tabel, dan indikator kinerja, untuk menampilkan informasi penting yang mendukung proses pemantauan, analisis, dan pengambilan keputusan. Berbeda dengan laporan statis, dashboard analitik bersifat dinamis

dan interaktif, sehingga memungkinkan pengguna untuk memperoleh gambaran kondisi secara cepat dan menyeluruh.

Visualisasi data memiliki peran penting dalam dashboard analitik karena mampu mengubah data mentah menjadi informasi yang lebih bermakna dan intuitif. Melalui visualisasi yang tepat, pengguna dapat dengan mudah mengidentifikasi pola, tren, serta perubahan yang terjadi pada data. Visualisasi data juga berfungsi sebagai sarana penyampaian informasi yang bersifat naratif, di mana kinerja dan kondisi bisnis dapat digambarkan secara komprehensif tanpa harus melalui proses analisis data yang kompleks.

Penelitian yang dilakukan oleh (Rathore et al., 2025) menunjukkan bahwa penerapan teknik visualisasi data yang efektif dapat membantu pengambil keputusan dalam menghasilkan keputusan yang lebih informatif, akurat, dan ringkas. Studi tersebut menegaskan bahwa dashboard, berbagai jenis grafik, serta pendekatan visualisasi yang terstruktur berperan penting dalam meningkatkan kualitas *business intelligence* dan mendukung perencanaan strategis. Selain itu, visualisasi data dinilai mampu meningkatkan efisiensi proses pengambilan keputusan dan mengurangi waktu yang dibutuhkan untuk menganalisis data, sehingga organisasi dapat merespons permasalahan dan dinamika bisnis secara lebih cepat.

Pentingnya analisis dan visualisasi data dalam konteks bisnis juga dikemukakan oleh (Purnama et al., 2025), yang menyatakan bahwa visualisasi data berperan dalam membantu organisasi memahami tren dan pola yang terdapat dalam data. Informasi yang dihasilkan dari proses analisis tersebut dapat dimanfaatkan untuk mendukung pengambilan keputusan serta perencanaan bisnis berbasis data.

Dalam konteks aplikasi data-driven, dashboard analitik berfungsi sebagai jembatan antara data dan pengambilan keputusan. Dashboard memungkinkan penyajian data yang informatif sehingga perubahan kondisi dan tren dapat dipantau secara berkelanjutan. Hal ini menjadikan dashboard analitik sebagai komponen penting dalam sistem yang memanfaatkan data berskala besar dan bersifat dinamis, termasuk data yang dihasilkan dari media sosial.

Bagi UMKM, keberadaan dashboard analitik menjadi sangat relevan karena dapat menyederhanakan informasi yang kompleks menjadi tampilan visual yang mudah dipahami. Informasi seperti kecenderungan sentimen konsumen, pola interaksi pengguna, dan ringkasan data pemasaran dapat disajikan secara visual, sehingga pelaku UMKM tidak perlu melakukan analisis data secara manual. Dengan demikian, dashboard analitik dapat mendukung UMKM dalam mengambil keputusan bisnis yang lebih cepat, tepat, dan berbasis data.

### **2.1.5 Blackbox Testing**

*Blackbox testing* merupakan salah satu metode pengujian perangkat lunak yang berfokus pada pengujian fungsional sistem dari sudut pandang pengguna tanpa memperhatikan struktur internal atau implementasi kode program. Pada metode ini, sistem diperlakukan sebagai sebuah “kotak hitam”, di mana pengujian dilakukan dengan memberikan input tertentu dan kemudian mengamati output atau respons yang dihasilkan untuk menilai kesesuaian fungsi sistem dengan spesifikasi yang telah ditetapkan. Pendekatan ini banyak digunakan dalam pengujian perangkat lunak karena mampu mengevaluasi perilaku sistem secara langsung berdasarkan kebutuhan pengguna.

Metode *blackbox testing* menitikberatkan pada validasi fungsi utama sistem, seperti proses input dan output, alur penggunaan, serta respons sistem terhadap berbagai kondisi penggunaan. Dengan demikian, metode ini sangat sesuai untuk menguji sistem berbasis antarmuka pengguna, khususnya aplikasi web dan dashboard, yang keberhasilannya sangat bergantung pada kesesuaian fungsi dan kemudahan penggunaan. Pengujian dilakukan tanpa melibatkan analisis terhadap logika internal atau struktur kode, sehingga hasil pengujian lebih berorientasi pada pengalaman dan kebutuhan pengguna akhir.

Penelitian yang dilakukan oleh Maulida et al. (Maulida et al., 2025) menunjukkan bahwa *blackbox testing* efektif digunakan dalam pengujian sistem website pemesanan online karena mampu mengidentifikasi kesalahan fungsional pada tahap awal pengembangan. Dalam penelitian tersebut, pengujian dilakukan pada berbagai fitur utama sistem, seperti proses registrasi, login, pencarian produk, hingga konfirmasi pesanan, tanpa meninjau kode program yang digunakan. Hasil penelitian tersebut membuktikan bahwa penerapan *blackbox testing* dapat membantu memastikan kualitas sistem dari aspek fungsionalitas sebelum sistem dirilis kepada pengguna.

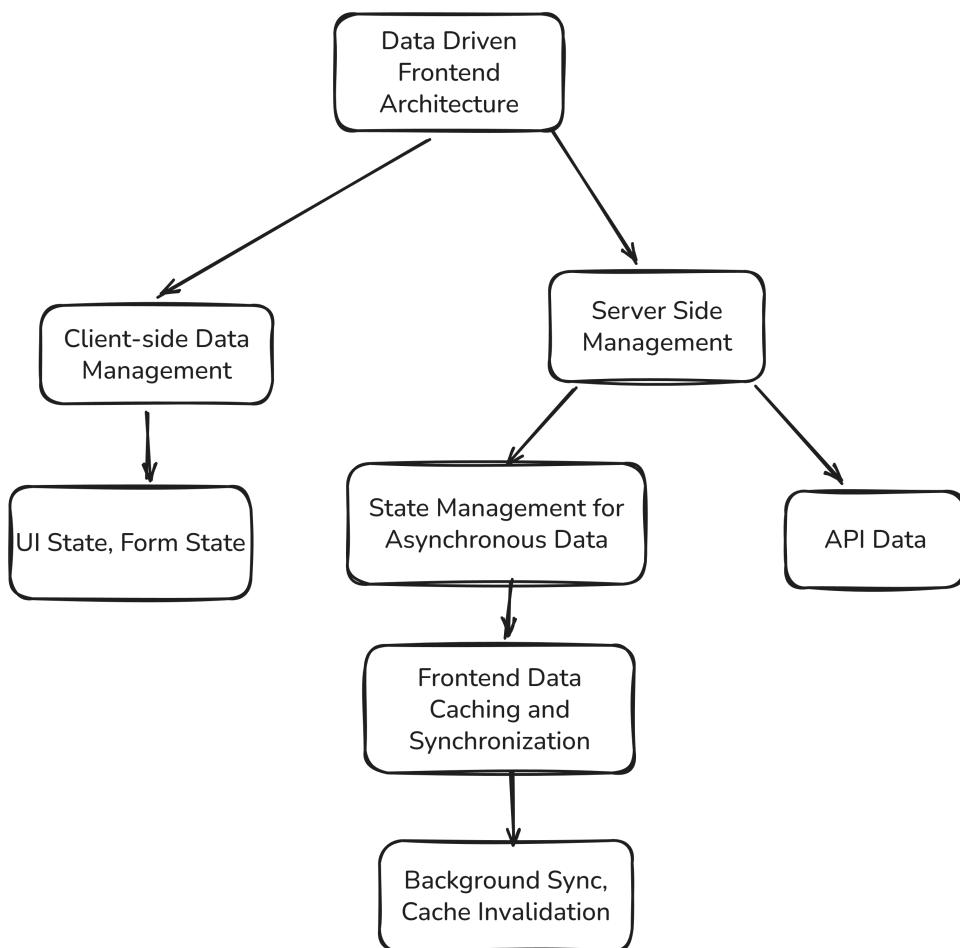
Berdasarkan karakteristik tersebut, black-box testing dipandang sebagai metode pengujian yang relevan untuk digunakan dalam penelitian ini. Metode ini memungkinkan evaluasi terhadap perilaku sistem *frontend* secara menyeluruh berdasarkan skenario penggunaan, sehingga kesesuaian fungsi sistem dengan kebutuhan pengguna dapat dinilai secara sistematis.

### **2.1.6 Arsitektur Client Data Layer**

Pada aplikasi *frontend* modern yang bersifat data-driven, pengelolaan data yang bersumber dari Application Programming Interface (API) menjadi salah satu aspek penting dalam arsitektur sistem. Aplikasi seperti dashboard analitik umumnya menampilkan berbagai komponen antarmuka, seperti grafik, tabel, dan indikator, yang bergantung pada data yang sama dan diperbarui berdasarkan sumber data dari API. Jika pengambilan dan pengolahan data

dilakukan secara langsung di setiap komponen antarmuka, maka dapat menimbulkan berbagai permasalahan, seperti permintaan API yang berulang, inkonsistensi data antar-komponen, serta peningkatan beban render yang berdampak pada penurunan performa aplikasi.

*Client Data Layer* merupakan pendekatan arsitektural dalam pengembangan *frontend* modern yang bertujuan untuk mengelola data yang bersumber dari server secara terpusat di sisi klien. Pendekatan ini muncul sebagai respons terhadap meningkatnya kompleksitas aplikasi data-driven, dimana data bersifat asinkron, dinamis, dan digunakan oleh banyak komponen antarmuka secara bersamaan. Dengan adanya *Client Data Layer*, proses *fetching*, *caching*, dan *synchronize data* dapat dilakukan secara lebih terstruktur, sehingga membantu menjaga konsistensi data dan mendukung pengelolaan data *frontend* secara lebih terstruktur.



**Gambar 2.1** Arsitektur *Client Data Layer*

### 1. *Data-driven Frontend Architecture*

*Data-driven Frontend Architecture* merupakan pendekatan pengembangan di mana seluruh antarmuka pengguna didorong oleh data sebagai sumber kebenaran utama. Dalam pola ini, UI dihasilkan sebagai fungsi dari state yang ada—artinya, setiap perubahan pada data akan secara otomatis memicu pembaruan pada tampilan aplikasi.

Arsitektur ini sangat berguna untuk aplikasi *frontend* yang menampilkan informasi dinamis seperti dashboard analitik, papan monitoring. Keunggulan utamanya adalah konsistensi tampilan yang lebih terjamin, alur data yang mudah dilacak, dan pengembangan fitur baru yang lebih sistematis karena UI dan logika data terpisah dengan jelas.

## 2. *Server State Management*

*Server state management* mengacu pada pengelolaan data yang bersumber dari sistem eksternal, seperti API atau layanan backend, yang bersifat asinkron dan dapat berubah di luar kendali langsung aplikasi klien. Data ini memiliki karakteristik dinamis karena dipengaruhi oleh kondisi jaringan, waktu respons server, serta pembaruan data di sisi backend. Oleh karena itu, server state memerlukan mekanisme khusus untuk menangani proses pengambilan data, status pemuatan, penanganan kesalahan, serta pembaruan dan sinkronisasi data agar informasi yang digunakan oleh berbagai komponen antarmuka tetap konsisten dan akurat.

## 3. *Client-side Data Management*

*Client-side data management* berfokus pada pengelolaan data di sisi klien setelah data tersebut diperoleh dari *server*, termasuk penyimpanan sementara, penggunaan ulang data, serta distribusi data ke berbagai komponen antarmuka. Pendekatan ini bertujuan untuk mengurangi ketergantungan terhadap permintaan data berulang ke server sekaligus menjaga konsistensi informasi yang ditampilkan pada antarmuka pengguna. Dengan pengelolaan data yang terstruktur di sisi klien, aplikasi *frontend* dapat menyajikan data secara lebih responsif dan stabil, khususnya pada aplikasi *frontend* yang bersifat data-driven seperti dashboard analitik, tanpa mencampurkan pengelolaan data server dengan state antarmuka pengguna.

## 4. *State Management for Asynchronous Data*

*State Management for Asynchronous Data* adalah pendekatan khusus untuk menangani data yang diperoleh melalui state, seperti panggilan API, operasi file, atau permintaan jaringan lainnya. Karena data tersebut tidak tersedia secara instan, sistem harus mampu mengelola berbagai state yang mungkin terjadi: mulai dari state *idle*, state *fetching* atau *onloading*, state *success*, hingga state *error*. Tantangan utamanya adalah memastikan aplikasi tetap responsif dan memberikan umpan balik yang informatif kepada pengguna selama proses pengambilan data. Pendekatan ini juga mencakup strategi seperti pembatalan permintaan yang tidak diperlukan, pengulangan otomatis saat gagal, dan pembaruan data latar belakang untuk menjaga

informasi tetap konsisten.

### 5. *Frontend Data Caching and Synchronization*

*Frontend Data Caching and Synchronization* adalah teknik untuk meningkatkan kinerja aplikasi dengan menyimpan salinan data dari server di memori klien. Dengan adanya *cache*, aplikasi dapat menampilkan informasi secara cepat tanpa perlu melakukan permintaan berulang ke server. Namun, teknik ini juga menimbulkan tantangan, yaitu data yang disimpan dapat menjadi kedaluwarsa jika terjadi perubahan di sisi server. Oleh karena itu, diperlukan mekanisme sinkronisasi yang cerdas, seperti pembaruan di latar belakang (background refresh) dan penandaan *cache* yang kedaluwarsa (cache invalidation). Dengan pendekatan ini, aplikasi dapat menampilkan data secara lebih konsisten sekaligus menjaga keakuratan informasi yang ditampilkan.

Beberapa penelitian menunjukkan bahwa penerapan mekanisme *caching* pada aplikasi *frontend* dapat meningkatkan performa dan responsivitas sistem. *Caching* memungkinkan data hasil pemanggilan API disimpan sementara di sisi klien sehingga dapat digunakan kembali tanpa melakukan permintaan ulang ke server. Pendekatan ini terbukti mampu mengurangi duplikasi permintaan API dan mempercepat waktu respons aplikasi. Pemanfaatan pustaka seperti TanStack Query dalam mengelola *caching* dan prefetching data juga dinilai efektif dalam meningkatkan efisiensi pengelolaan data di sisi *frontend* serta pengalaman pengguna secara keseluruhan (Rahman and Prihanto, 2024). Dalam konteks penelitian ini, peningkatan performa dipahami sebagai perubahan mekanisme pengelolaan data *frontend* yang diamati melalui mekanisme *caching* dan pengendalian permintaan data, tanpa dilakukan pengukuran performa numerik.

Melalui fitur-fitur seperti *smart caching*, *refetching*, dan dukungan terhadap penanganan data asinkron, pendekatan ini membantu menyederhanakan kompleksitas pengelolaan data yang bersumber dari API serta mendukung peningkatan efisiensi pengelolaan data di sisi frontend. Selain itu, dukungan terhadap TypeScript memungkinkan konsistensi struktur data tetap terjaga tanpa mencampurkan pengelolaan server state dengan *client state*, sehingga sesuai dengan kebutuhan aplikasi *frontend* modern yang bersifat data-driven (Highlights, 2025).

Perlu dibedakan secara konseptual antara *client state* dan *server state* dalam pengembangan aplikasi *frontend* modern. *Client state* mengacu pada state yang sepenuhnya berada di sisi klien dan berkaitan langsung dengan antarmuka pengguna, seperti status tampilan,

input formulir, filter, atau kondisi navigasi. *Server state* merepresentasikan data yang bersumber dari sistem eksternal seperti API atau layanan backend, bersifat asinkron, dapat berubah di luar kendali aplikasi klien, serta memerlukan penanganan khusus terkait proses pengambilan data, status pemuatan, penanganan kesalahan, dan sinkronisasi.

Dalam konteks pengembangan aplikasi frontend yang bersifat data-driven, istilah data frontend terstruktur pada penelitian ini merujuk pada cara data yang bersumber dari server dikelola, didistribusikan, dan digunakan di sisi klien. Data yang dikelola melalui mekanisme terpusat di sisi *frontend* memiliki bentuk dan siklus hidup yang konsisten, serta dapat digunakan ulang oleh berbagai komponen antarmuka tanpa menimbulkan inkonsistensi informasi maupun permintaan data yang tidak terkontrol. Definisi ini selaras dengan prinsip *frontend state management* yang dikemukakan oleh (Evergreen, 2023), di mana keterstrukturan data ditentukan oleh bagaimana data dikendalikan, dipertahankan, dan disinkronkan di dalam aplikasi.

Berdasarkan definisi tersebut, pengelolaan data *frontend* dapat dikatakan terstruktur apabila memenuhi beberapa kriteria berikut:

1. Pengelolaan Data Terpusat (*Single Source of Truth*)

Data yang bersumber dari server dikelola melalui satu lapisan pengelolaan data di sisi *frontend*, sehingga komponen antarmuka tidak melakukan pengambilan data secara langsung ke API. Pendekatan ini menghindari duplikasi data dan memastikan seluruh komponen menggunakan sumber data yang sama, sehingga menciptakan prediksi yang lebih baik terhadap perubahan state (Evergreen, 2023).

2. Konsistensi Bentuk dan Representasi Data

Data yang digunakan oleh komponen antarmuka memiliki struktur dan representasi yang konsisten, baik dari segi skema, *interface* maupun nilai yang ditampilkan. Dengan konsistensi ini, perbedaan tampilan informasi antar-komponen dapat dihindari meskipun data digunakan pada konteks visualisasi yang berbeda.

3. Siklus Hidup Data

Data *frontend* memiliki status yang terdefinisi dengan jelas, seperti *loading*, *success*, dan *error*. Kejelasan siklus hidup ini memungkinkan sistem untuk memberikan umpan balik yang tepat kepada pengguna serta memudahkan pengelolaan data yang bersifat asinkron, termasuk klasifikasi state berdasarkan umur panjangnya (*persistent* vs *non-persistent*) (Evergreen, 2023).

4. Pemanfaatan Mekanisme *Caching*

Data hasil pengambilan dari server disimpan sementara di sisi frontend dan dapat digunakan kembali oleh komponen lain tanpa harus melakukan permintaan ulang

ke API. Mekanisme *caching* berperan penting dalam mengurangi permintaan data yang berulang serta meningkatkan efisiensi pengelolaan data, sekaligus mendukung persistensi data untuk menghindari kehilangan informasi saat reload (Evergreen, 2023).

#### 5. Sinkronisasi Data Antar-Komponen

Perubahan data yang terjadi akibat pembaruan atau refetch dapat tercermin secara otomatis pada seluruh komponen yang menggunakan data tersebut. Dengan adanya sinkronisasi, informasi yang ditampilkan pada dashboard tetap konsisten meskipun terjadi interaksi pengguna atau navigasi antar-halaman, sehingga mengoptimalkan performa seperti minimisasi re-render yang tidak perlu (Evergreen, 2023).

#### 6. *Readability* (Keterbacaan Struktur)

Sistem pengelolaan data harus terorganisir dengan baik untuk memudahkan pemeliharaan dan pengembangan, menghindari redundansi data atau modifikasi state yang tidak perlu, sehingga meningkatkan kemudahan kolaborasi antar-komponen (Evergreen, 2023).

#### 7. *Scalability* (Skalabilitas)

Pengelolaan data harus mendukung pertumbuhan aplikasi dengan menangani peningkatan volume data dan state secara efisien, tanpa membatasi ekstensi fitur atau menimbulkan bottleneck performa (Evergreen, 2023).

Pengelolaan data *frontend* yang terstruktur menuntut adanya mekanisme pengelolaan *server state* yang mampu mengatur proses pengambilan, penyimpanan sementara, dan pembaruan data secara konsisten. Dalam konteks ini, pemanfaatan mekanisme *caching* memungkinkan data yang diperoleh dari server digunakan ulang oleh berbagai komponen tanpa menimbulkan permintaan API berulang, sekaligus tetap mendukung pembaruan data ketika diperlukan. Selain itu, pengelolaan data secara terpusat dan adanya mekanisme sinkronisasi memastikan konsistensi data antar-komponen, sehingga seluruh antarmuka menampilkan informasi yang selaras. Dengan demikian, keterstrukturkan data *frontend* dalam penelitian ini dipahami sebagai integrasi antara pengelolaan *server state*, penerapan *caching*, dan sinkronisasi data antar-komponen, yang menjadi fondasi penerapan arsitektur *Client Data Layer* untuk menghasilkan pengelolaan data *frontend* yang konsisten, dan efisien.

Pengelolaan data *frontend* yang tidak terstruktur terjadi ketika data yang bersumber dari server dikelola secara langsung oleh masing-masing komponen antarmuka tanpa mekanisme pengelolaan terpusat di sisi frontend. Pada pendekatan ini, proses pengambilan, penyimpanan, dan pembaruan data dilakukan secara terpisah di setiap komponen. Akibatnya, data tidak

memiliki siklus hidup yang konsisten, seperti status loading, success, dan error dikelola secara terpisah, serta data yang sama berpotensi diambil ulang dari API dan tidak adanya mekanisme *caching*.

Kondisi tersebut berdampak langsung pada inkonsistensi data antar-komponen, karena setiap komponen dapat menampilkan data pada waktu dan kondisi pembaruan yang berbeda. Pada aplikasi dashboard analitik, hal ini berpotensi menimbulkan perbedaan informasi antar-komponen atau antar-halaman dan mengurangi tingkat kepercayaan terhadap sistem sebagai alat bantu pengambilan keputusan berbasis data. Oleh karena itu, keterbatasan pengelolaan data langsung di tingkat komponen menjadi dasar perlunya penerapan arsitektur *Client Data Layer* untuk mencapai pengelolaan data *frontend* yang lebih terstruktur, konsisten, dan efisien.

### 2.1.7 REST API

*Representational State Transfer Application Programming Interface (REST API)* merupakan gaya arsitektur layanan web yang digunakan untuk memungkinkan komunikasi antara klien dan server melalui protokol HTTP secara terstandarisasi. REST API bersifat stateless, di mana setiap permintaan dari klien harus membawa seluruh informasi yang dibutuhkan untuk diproses oleh server, sehingga tidak bergantung pada status permintaan sebelumnya. Data yang dipertukarkan umumnya disajikan dalam format JSON karena bersifat ringan dan mudah diproses oleh aplikasi *frontend*, menjadikan REST API banyak digunakan pada aplikasi web modern yang bersifat data-driven.

Dalam konteks aplikasi *frontend analitik*, REST API berperan sebagai sumber utama data (server state) yang dikonsumsi oleh antarmuka pengguna. Data yang diperoleh melalui REST API bersifat asinkron dan dapat berubah sewaktu-waktu, sehingga memerlukan mekanisme pengelolaan data yang mampu menangani proses pengambilan, pembaruan, dan sinkronisasi data secara efisien. Penelitian terkini menunjukkan bahwa REST API memiliki keunggulan dalam penyajian data yang bersifat datar dan mudah di-cache, sehingga mampu meningkatkan efisiensi distribusi data serta memperbesar rasio cache hit pada lapisan jaringan. Pendekatan ini dinilai lebih optimal untuk kebutuhan aplikasi yang menampilkan data terstruktur secara berulang, seperti dashboard dan sistem pelaporan, dibandingkan pendekatan API lain yang lebih kompleks (Islam, 2025).

Lebih lanjut, penelitian tersebut menegaskan bahwa performa aplikasi web tidak ditentukan oleh satu teknologi tertentu, melainkan oleh keselarasan antara desain akses data, mekanisme *caching*, serta pengelolaan state pada sisi klien. REST API yang dirancang dengan kontrak data yang jelas dan *cache-aware* terbukti mendukung peningkatan performa dan skalabilitas aplikasi ketika dipadukan dengan lapisan pengelolaan data di *frontend*. Oleh karena

itu, dalam pengembangan aplikasi *frontend* modern, REST API umumnya tidak diakses secara langsung oleh setiap komponen antarmuka, melainkan melalui lapisan pengelolaan data seperti *Client Data Layer* agar data dapat dikelola secara terpusat, konsisten, dan efisien.

Dalam penelitian ini, REST API diposisikan sebagai penyedia data hasil analisis sentimen yang diproses di sisi backend. Data tersebut selanjutnya dikelola pada sisi frontend melalui arsitektur *Client Data Layer* sebelum ditampilkan dalam bentuk visualisasi pada dashboard analitik. Dengan pemisahan peran ini, REST API berfungsi sebagai sumber data, sementara pengelolaan performa, *caching*, dan sinkronisasi data dilakukan sepenuhnya di sisi *frontend* untuk mendukung penyajian informasi yang responsif dan konsisten.

Penerapan *Client Data Layer* memberikan sejumlah manfaat dalam pengembangan aplikasi *frontend*. Salah satu manfaat utama adalah peningkatan konsistensi data, di mana beberapa komponen yang membutuhkan data yang sama dapat memperoleh informasi yang seragam tanpa harus melakukan permintaan data secara terpisah. Selain itu, *Client Data Layer* memungkinkan pengurangan jumlah permintaan API yang tidak diperlukan melalui mekanisme caching dan pengelolaan siklus data. Pendekatan ini juga mendukung penyajian data yang lebih stabil dan terkelola serta mempermudah pengelolaan data yang bersifat asinkron dan dinamis.

Dalam konteks dashboard analitik, keberadaan *Client Data Layer* menjadi semakin penting karena data yang ditampilkan umumnya bersifat besar, sering diperbarui, dan digunakan oleh banyak komponen secara bersamaan. Dengan memanfaatkan *Client Data Layer*, dashboard dapat menampilkan data secara lebih responsif dan stabil, sekaligus meminimalkan risiko inkonsistensi informasi yang ditampilkan kepada pengguna. Pendekatan ini mendukung terciptanya arsitektur *frontend* yang lebih terorganisasi, mudah dipelihara, dan skalabel.

### 2.1.8 React

React merupakan sebuah pustaka (*library*) JavaScript yang digunakan untuk membangun antarmuka pengguna (user interface) pada aplikasi web yang bersifat interaktif dan responsif. Menurut dokumentasi resmi React, pustaka ini dirancang untuk membangun antarmuka dengan pendekatan *component-based*, di mana tampilan antarmuka dibagi menjadi bagian-bagian kecil yang dapat digunakan kembali (*reusable components*) sehingga memudahkan pengelolaan dan pemeliharaan kode aplikasi (React, 2026). React bekerja dengan memanfaatkan virtual DOM untuk meminimalkan operasi pada DOM aktual sehingga perubahan tampilan dapat dilakukan secara efisien saat data atau state aplikasi berubah, tanpa perlu melakukan refresh seluruh halaman.

Pendekatan *component-based architecture* yang digunakan React juga memungkinkan pengembang merancang antarmuka sebagai susunan komponen modular yang saling terpisah

namun saling berinteraksi, yang selaras dengan prinsip pengembangan frontend modern. Struktur seperti ini tidak hanya meningkatkan keterbacaan dan modularitas kode, tetapi juga memberikan fleksibilitas dalam pengembangan aplikasi berskala besar serta mempermudah pengujian dan pemeliharaan. Dalam konteks pengembangan aplikasi *Single Page Application* (*SPA*) dan *API-driven application*, studi yang dilakukan pada tren pengembangan aplikasi web menunjukkan bahwa penggunaan React dalam kombinasi dengan arsitektur API terbukti mendukung pengembangan aplikasi web yang modular, fleksibel, dan mudah dikembangkan secara berkelanjutan (Putra et al., 2025).

Dengan karakteristik tersebut, React dipilih sebagai teknologi *frontend* dalam penelitian ini untuk membangun antarmuka pengguna yang dinamis dan dapat berinteraksi secara langsung dengan lapisan pengelolaan data (*Client Data Layer*), sehingga mendukung kebutuhan aplikasi yang bersifat *data-driven*.

### 2.1.9 TanStack Query (React Query)

TanStack Query merupakan pustaka manajemen data pada sisi *frontend* yang dirancang untuk mengelola data yang bersumber dari server (*server state*) secara efisien. Dalam dokumentasi resminya, TanStack Query dijelaskan sebagai "*the missing data-fetching layer for web applications*" yang berfungsi untuk mempermudah proses pengambilan, penyimpanan sementara (*caching*), sinkronisasi, serta pembaruan data dari server (Tanstack LCC, 2025). Pendekatan ini ditujukan untuk menangani kompleksitas data asinkron yang tidak dapat dikelola secara optimal menggunakan mekanisme state management konvensional.

Dalam dokumentasi resminya, TanStack Query mendefinisikan server state sebagai data yang berasal dari sumber eksternal dan memiliki karakteristik asinkron, dapat berubah sewaktu-waktu, serta memerlukan mekanisme khusus untuk menjaga konsistensi data di sisi klien. Oleh karena itu, TanStack Query menyediakan pendekatan deklaratif dalam pengelolaan server state, di mana pengembang dapat mendefinisikan kebutuhan data tanpa harus menangani secara manual proses sinkronisasi dan pembaruan data di setiap komponen antarmuka (TanStack Documentation, 2024).

Salah satu fitur utama TanStack Query adalah mekanisme *caching* yang memungkinkan data hasil pemanggilan API disimpan sementara di sisi klien. Dengan adanya *caching*, data yang telah diperoleh dapat digunakan kembali oleh komponen lain tanpa perlu melakukan permintaan ulang ke server, selama data tersebut masih dianggap valid. Pendekatan ini berkontribusi dalam mengurangi jumlah permintaan API yang tidak diperlukan, meningkatkan efisiensi aplikasi, serta mempercepat waktu respons antarmuka pengguna.

Selain *caching*, TanStack Query juga menyediakan mekanisme sinkronisasi data yang

mendukung pembaruan data secara otomatis. Melalui konsep seperti refetching dan invalidasi data, TanStack Query memastikan bahwa data yang ditampilkan tetap mutakhir ketika terjadi perubahan di sisi server. Mekanisme ini sangat relevan pada aplikasi data-driven, seperti dashboard analitik, yang menampilkan data secara dinamis dan digunakan oleh banyak komponen secara bersamaan.

Dalam konteks arsitektur *frontend*, TanStack Query dapat diposisikan sebagai implementasi konkret dari *Client Data Layer*. Pustaka ini berperan sebagai lapisan perantara antara backend API dan komponen antarmuka pengguna, sehingga komponen UI tidak berinteraksi langsung dengan API. Dengan demikian, TanStack Query membantu memisahkan logika pengelolaan data dari logika tampilan, meningkatkan keterbacaan kode, serta mempermudah pemeliharaan aplikasi dalam jangka panjang.

Pada penelitian ini, TanStack Query digunakan sebagai solusi untuk menerapkan arsitektur *Client Data Layer* pada pengembangan dashboard analitik. Fokus penggunaan TanStack Query diarahkan pada pengelolaan server state, caching data, serta sinkronisasi data antar-komponen, tanpa membahas aspek internal pustaka atau detail implementasi secara mendalam. Dengan pendekatan ini, TanStack Query berperan sebagai fondasi pengelolaan data pada sisi *frontend* yang mendukung penyajian informasi sentimen secara konsisten, responsif, dan efisien.

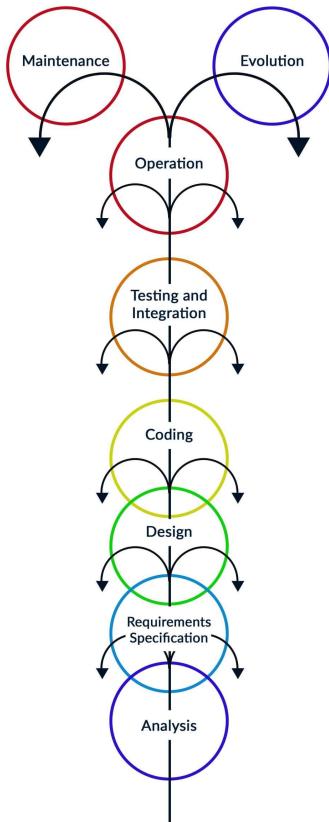
### 2.1.10 Metode Fountain

Metode Fountain merupakan salah satu model dalam *Software Development Life Cycle (SDLC)* yang bersifat iteratif dan fleksibel, sebagai alternatif terhadap model pengembangan linear seperti Waterfall. Model ini memungkinkan fase-fase pengembangan seperti analisis kebutuhan, perancangan, implementasi, dan pengujian untuk saling tumpang tindih serta dilakukan ulang sesuai kebutuhan proyek, sehingga proses pengembangan dapat menyesuaikan perubahan persyaratan yang muncul sepanjang siklus hidup sistem.

Selain bersifat iteratif, metode Fountain menekankan kelanjutan antar-tahapan pengembangan tanpa batasan transisi yang kaku. Setiap aktivitas dalam siklus pengembangan tidak dipandang sebagai tahap yang berdiri sendiri. Dengan pendekatan ini, umpan balik yang diperoleh pada tahap implementasi atau pengujian dapat segera digunakan untuk memperbaiki hasil analisis maupun perancangan sebelumnya. Hal tersebut memungkinkan kesalahan desain atau ketidaksesuaian kebutuhan terdeteksi lebih awal, sehingga risiko *rework* pada tahap akhir dapat diminimalkan.

Fleksibilitas metode Fountain menjadikannya relevan untuk pengembangan sistem yang bersifat dinamis, seperti aplikasi dashboard analitik. Pada sistem semacam ini, perubahan

kebutuhan pengguna, penyesuaian arsitektur sering kali muncul selama proses pengembangan berlangsung. Oleh karena itu, model Fountain memberikan kerangka kerja yang adaptif namun tetap terstruktur, sehingga pengembangan sistem dapat berjalan secara terkendali tanpa kehilangan kemampuan untuk beradaptasi terhadap perubahan yang terjadi.



**Gambar 2.2** Fountain SDLC Model

Gambar 2.2 menunjukkan ilustrasi model Fountain, di mana fase-fase pengembangan tidak lagi bergerak secara kaku dari satu tahap ke tahap berikutnya, tetapi fase dapat saling berinteraksi dan kembali ke fase sebelumnya bila terdapat kebutuhan penyesuaian. Karakteristik ini memungkinkan proses pengembangan sistem menjadi lebih adaptif terhadap perubahan yang tidak terduga, sehingga risikonya dapat diminimalkan dan kualitas akhir sistem meningkat.

Fountain banyak digunakan pada proyek yang memerlukan peninjauan ulang fase secara berkala, terutama ketika kebutuhan belum sepenuhnya jelas di awal atau kemungkinan perubahan tinggi. Dengan mekanisme iteratif dan fleksibel, model ini mendukung evaluasi berkelanjutan terhadap artefak pengembangan tanpa mengganggu keseluruhan proses pengembangan sistem.

Menurut (Siva et al., 2023), metode Fountain banyak digunakan pada pengembangan perangkat lunak dengan kompleksitas menengah hingga tinggi, khususnya pada sistem yang kebutuhan fungsionalnya dapat berkembang seiring berjalannya proses implementasi. Dengan

sifatnya yang fleksibel dan iteratif, metode ini dinilai sesuai untuk penelitian rekayasa perangkat lunak yang membutuhkan penyesuaian desain dan pengambilan keputusan teknis secara berulang tanpa mengganggu keseluruhan alur pengembangan sistem .

### Tahapan Metode Fountain

Metode Fountain terdiri dari beberapa tahapan pengembangan perangkat lunak yang dilakukan secara iteratif dan fleksibel. Setiap tahapan tidak bersifat kaku dan dapat saling tumpang tindih, sehingga memungkinkan penyesuaian kembali ke tahapan sebelumnya apabila ditemukan perubahan kebutuhan atau permasalahan selama proses pengembangan. Secara umum, tahapan dalam metode Fountain meliputi analisis, requirement specification, design, coding, testing, operation, maintenance, dan Evolution Siva et al. (2023).

Tahap *analysis* merupakan tahapan awal yang bertujuan untuk memahami permasalahan dan kebutuhan sistem secara umum. Pada tahap ini dilakukan identifikasi kebutuhan pengguna, ruang lingkup sistem, serta permasalahan yang ingin diselesaikan melalui pengembangan perangkat lunak. Hasil dari tahap analisis menjadi dasar bagi tahapan-tahapan selanjutnya.

Tahap *requirement specification* berfokus pada perumusan kebutuhan sistem secara lebih terstruktur dan terdokumentasi. Kebutuhan tersebut mencakup kebutuhan fungsional dan non-fungsional yang harus dipenuhi oleh sistem. Spesifikasi kebutuhan berperan penting sebagai acuan dalam proses perancangan dan implementasi sistem.

Tahap *design* merupakan tahapan perancangan solusi berdasarkan spesifikasi kebutuhan yang telah ditetapkan. Pada tahap ini dilakukan perancangan arsitektur sistem, struktur data, serta rancangan antarmuka pengguna. Hasil perancangan bertujuan untuk memberikan gambaran teknis mengenai bagaimana sistem akan dibangun sebelum masuk ke tahap implementasi.

Tahap *coding* adalah tahapan implementasi dari desain yang telah dibuat ke dalam bentuk kode program. Pada tahap ini, pengembang menerjemahkan rancangan sistem menjadi perangkat lunak yang dapat dijalankan sesuai dengan kebutuhan yang telah ditentukan.

Tahap *testing* dilakukan untuk memastikan bahwa sistem yang dikembangkan telah berjalan sesuai dengan spesifikasi dan bebas dari kesalahan fungsional. Pengujian dilakukan untuk memverifikasi fungsi-fungsi sistem serta memastikan bahwa sistem dapat digunakan dengan baik oleh pengguna.

Tahap *operation* merupakan tahapan di mana sistem telah siap digunakan dalam lingkungan operasional. Pada tahap ini, sistem mulai diakses oleh pengguna dan digunakan untuk mendukung aktivitas yang telah dirancang.

Tahap *maintenance* bertujuan untuk menjaga kinerja sistem setelah digunakan secara operasional. Aktivitas pada tahap ini meliputi perbaikan kesalahan, penyesuaian terhadap

perubahan lingkungan, serta peningkatan minor terhadap sistem agar tetap berjalan dengan optimal.

Tahap *Evolution* merupakan tahapan pengembangan lanjutan dalam metode Fountain yang bertujuan untuk melakukan peningkatan dan penambahan fitur pada sistem berdasarkan hasil penggunaan dan kebutuhan yang muncul. Tahapan ini memastikan sistem dapat terus dikembangkan secara berkelanjutan agar tetap sesuai dengan kebutuhan pengguna dan tujuan pengembangan

### **Alasan Pemilihan Metode Fountain**

Pemilihan metode *Software Development Life Cycle (SDLC)* yang tepat merupakan langkah strategis yang krusial untuk memastikan keberhasilan proyek serta mencegah pembengkakan biaya dan waktu pengembangan (Aniley et al., 2024). Berdasarkan pertimbangan tersebut, penelitian ini mengadopsi metode Fountain yang dinilai relevan karena memiliki karakteristik fleksibel namun tetap terstruktur. Efektivitas metode ini didukung oleh penelitian (Sastra and Sutawinata, 2023) pada perancangan sistem SIP-PTK, yang menunjukkan bahwa model Fountain mampu memandu tahapan analisis, desain, implementasi, hingga pengujian secara efektif pada sistem yang memiliki kebutuhan pengolahan data yang spesifik.

Relevansi metode Fountain juga sejalan dengan fokus penelitian ini, yaitu pengembangan aplikasi *frontend* yang bersifat data-driven dan memiliki ketergantungan tinggi terhadap interaksi antar-komponen. Dalam pengembangan *frontend*, tahapan perancangan arsitektur, implementasi, dan pengujian tidak berjalan secara linier, melainkan saling berkaitan dan sering memerlukan penyesuaian berdasarkan hasil evaluasi sistem. Penerapan metode Fountain memungkinkan tahapan pengembangan dan evaluasi berjalan secara paralel serta saling tumpang tindih (overlapping) tanpa menghilangkan struktur dan urutan penelitian yang jelas.

Melalui pendekatan ini, aktivitas analisis kebutuhan, perancangan arsitektur *frontend*, implementasi *Client Data Layer*, serta pengujian sistem dapat dilakukan secara berulang dan saling mempengaruhi. Hasil evaluasi pada satu tahapan dapat langsung digunakan sebagai dasar penyesuaian pada tahapan lainnya, seperti perubahan struktur data atau mekanisme sinkronisasi, tanpa harus menunggu seluruh siklus pengembangan selesai. Dengan demikian, proses pengembangan sistem menjadi lebih adaptif dan terkontrol, sehingga diharapkan mampu menghasilkan hasil penelitian yang sesuai dengan tujuan penelitian secara optimal.

## **2.2 Penelitian Terkait**

Berikut adalah tabel perbandingan penelitian terkait yang relevan dengan pengembangan penerapan arsitektur *Client Data Layer* menggunakan TanStack Query pada dashboard analisis sentimen

**Tabel 2.1** Tabel Perbandingan Penelitian Terkait

No	Peneliti	Teknologi	Judul	Fitur
1	Fajarini, Sri Dwi; Kurniawati, Juliana; Yuliani, Fitria (2025)	NLP, Machine Learning (SVM, Random Forest, VADER)	<i>Social Media Sentiment Analysis as a New Tool for Predicting Market Trends and Consumer Behaviour</i>	Analisis sentimen media sosial untuk mengidentifikasi pola opini publik dan memprediksi perilaku konsumen serta tren pasar.
2	Shrutika Rathore; Rahul Nawkhare; Navin Sharma; Nitin Chaudhary; Saurabh Chakole; Bhaskar Vishwakrama (2025)	Dashboard analitik, visualisasi data, business intelligence	<i>Effective Data Visualization Techniques for Business Decision-Makers</i>	Penyajian data bisnis melalui dashboard analitik untuk meningkatkan efisiensi pengambilan keputusan dan perencanaan strategis.
3	Micheal, Dave (2024)	React, TanStack Query, lazy loading, caching	<i>React Query and Lazy Loading: Performance Optimization Best Practices</i>	Optimasi performa aplikasi <i>frontend</i> melalui pengelolaan data asinkron, caching, dan lazy loading untuk mengurangi permintaan API berulang.

Berdasarkan kajian terhadap jurnal-jurnal penelitian terdahulu, dapat diidentifikasi adanya celah gap penelitian yang berkaitan dengan pemanfaatan hasil analisis sentimen media sosial pada sisi *frontend* aplikasi dashboard analitik. Sejumlah penelitian berfokus pada penerapan analisis sentimen menggunakan pendekatan Natural Language Processing (NLP) dan machine learning untuk memprediksi perilaku konsumen serta tren pasar. Namun demikian, penelitian-penelitian

tersebut umumnya menempatkan analisis sentimen sebagai fokus utama dan belum membahas secara mendalam bagaimana hasil analisis sentimen tersebut dikelola dan disajikan pada sisi *frontend*, khususnya dalam bentuk dashboard analitik yang digunakan secara langsung oleh pengguna.

Selain itu, penelitian lain menekankan pada peran dashboard dan teknik visualisasi data dalam meningkatkan efektivitas pengambilan keputusan bisnis. Fokus kajian lebih diarahkan pada desain visualisasi, jenis grafik, serta manfaat dashboard sebagai alat bantu analisis. Akan tetapi, aspek teknis pengelolaan data pada sisi *frontend*, seperti arsitektur pengambilan data dari API, pengelolaan server state, mekanisme caching, serta konsistensi data antar-komponen pada aplikasi *frontend* modern, masih belum menjadi perhatian utama dalam penelitian-penelitian tersebut.

Di sisi lain, terdapat penelitian yang membahas penggunaan React Query dalam pengelolaan data asinkron pada aplikasi React, termasuk pemanfaatan caching dan lazy loading untuk meningkatkan performa aplikasi *frontend*. Meskipun penelitian ini menunjukkan efektivitas React Query dalam mengurangi pemanggilan API berulang, konteks penerapannya masih bersifat umum dan belum secara spesifik dikaji sebagai bagian dari arsitektur *Client Data Layer* pada aplikasi dashboard analisis sentimen. Selain itu, keterkaitan antara pengelolaan server state di sisi *frontend* dengan kebutuhan penyajian data analitik pada konteks UMKM juga belum banyak dibahas. Oleh karena itu, penelitian ini mengisi celah tersebut dengan mengkaji penerapan arsitektur *Client Data Layer* menggunakan TanStack Query pada pengembangan Dashboard Analisis Sentimen UMKM.

*– Halaman ini sengaja dikosongkan –*

## **BAB 3**

### **METODOLOGI PENELITIAN**

#### **3.1 Waktu dan Jadwal penelitian**

##### **3.1.1 Waktu Pelaksanaan Penelitian**

Waktu pelaksanaan penelitian ini direncanakan selama 6 bulan, yaitu dimulai pada bulan Agustus 2025 dan berakhir pada bulan Januari 2026. Rentang waktu tersebut dipilih untuk memastikan seluruh tahapan penelitian dapat dilaksanakan secara sistematis dan terstruktur, mulai dari analisis kebutuhan, perancangan sistem, implementasi, hingga pengujian. Pembagian waktu penelitian disusun secara bertahap agar setiap aktivitas penelitian dapat dilakukan secara optimal sesuai dengan metode yang digunakan, serta memberikan ruang untuk penyesuaian apabila ditemukan kendala selama proses pengembangan sistem.

##### **3.1.2 Jadwal Kegiatan Penelitian**

Berikut adalah serangkaian jadwal kegiatan yang dilakukan selama pelaksanaan penelitian ini, yang diuraikan pada Tabel 3.1.

**Tabel 3.1** Jadwal Pelaksanaan Penelitian

No	Nama Kegiatan	Waktu Pelaksanaan					
		2025					2026
		Ags	Sep	Okt	Nov	Des	Jan
1	Analisis Kebutuhan Sistem						
2	Perumusan Ide dan Studi Awal Permasalahan						
3	Penyusunan Proposal dan Konsep Solusi						

No	Nama Kegiatan	Waktu Pelaksanaan					
		2025					2026
		Ags	Sep	Okt	Nov	Des	Jan
4	Pengumpulan Proposal dan Video Presentasi						
5	Seleksi Proposal dan Penetapan Finalis						
6	Presentasi dan Penjurian Akhir						
7	Pengembangan Arsitektur <i>Frontend</i> dan <i>Client Data Layer</i>						
8	Implementasi Dashboard dan TanStack Query						
9	Pengujian Sistem (Black-box Testing)						
10	Penyusunan Laporan Tugas Akhir						

Berdasarkan jadwal kegiatan penelitian yang telah disusun, pelaksanaan penelitian diawali pada bulan Agustus 2025 dengan tahap analisis kebutuhan sistem. Pada tahap ini dilakukan identifikasi permasalahan, pengumpulan kebutuhan pengguna, serta menyusun kebutuhan terhadap data dan API yang digunakan sebagai sumber data dashboard analisis sentimen. Tahap analisis ini menjadi dasar bagi tahapan penelitian selanjutnya. Pada bulan September 2025, kegiatan penelitian difokuskan pada perancangan sistem, yang meliputi perancangan arsitektur *Frontend*, perancangan *Client Data Layer*, serta perancangan antarmuka pengguna (UI). Tahap

perancangan bertujuan untuk menghasilkan rancangan sistem yang terstruktur dan sesuai dengan kebutuhan yang telah dianalisis sebelumnya. Pada periode bulan September hingga Desember 2025, kegiatan penelitian difokuskan pada tahapan perancangan dan implementasi sistem yang dilakukan secara iteratif dan saling tumpang tindih. Pada periode ini, proses perancangan antarmuka pengguna (desain antarmuka), perancangan sistem, perancangan arsitektur *Frontend* dan *Client Data Layer*, serta implementasi *Frontend* menggunakan TanStack Query tidak dilakukan secara terpisah dan linier, melainkan berjalan secara bersamaan sesuai dengan karakteristik metode Fountain yang fleksibel. Pendekatan ini memungkinkan hasil dari tahap implementasi *Frontend* untuk secara langsung digunakan sebagai dasar penyesuaian pada tahap perancangan sistem maupun perancangan arsitektur *Frontend*. Sebaliknya, perubahan pada desain antarmuka atau struktur arsitektur juga dapat segera diimplementasikan dan diuji tanpa harus menunggu selesainya seluruh tahapan sebelumnya. Dengan demikian, proses pengembangan sistem dapat berjalan lebih adaptif terhadap kebutuhan dan temuan selama penelitian. Pelaksanaan tahapan-tahapan tersebut secara paralel bertujuan untuk menjaga konsistensi antara rancangan arsitektur, mekanisme pengelolaan data pada *Client Data Layer*, serta implementasi antarmuka pengguna. Pola kerja ini sejalan dengan prinsip metode Fountain yang memungkinkan terjadinya pengulangan dan penyesuaian antar-tahapan pengembangan tanpa mengganggu keseluruhan alur penelitian. Selain itu, pendekatan paralel ini memberikan ruang bagi peneliti untuk melakukan evaluasi teknis secara berkelanjutan terhadap mekanisme pengelolaan data *Frontend*, khususnya terkait konsistensi data dan pengendalian pemanggilan API. Dengan adanya evaluasi berulang tersebut, setiap penyesuaian yang dilakukan dapat langsung divalidasi kesesuainya terhadap tujuan penelitian.

### 3.2 Metode Fountain

Berdasarkan landasan teori dan pembahasan metode penelitian yang telah diuraikan pada Bab II, penelitian ini menggunakan metode Fountain sebagai pendekatan dalam pengembangan sistem. Metode Fountain dipilih karena memiliki karakteristik fleksibel dan iteratif, sehingga memungkinkan tahapan analisis, perancangan, implementasi, dan pengujian dilakukan secara saling tumpang tindih sesuai dengan kebutuhan penelitian. Karakteristik tersebut dinilai sesuai dengan pengembangan aplikasi *frontend* yang bersifat data-driven dan memerlukan evaluasi berulang terhadap arsitektur dan pengelolaan data. Pada Bab III ini, metode Fountain diterapkan secara sistematis pada penelitian yang dilakukan. Pembahasan difokuskan pada tahapan penerapan metode Fountain dalam konteks pengembangan Dashboard Analisis Sentimen, mulai dari tahap analisis hingga tahap testing. Uraian pada setiap tahapan menjelaskan aktivitas yang dilakukan dalam penelitian ini tanpa mengulang pembahasan teoritis yang telah

disampaikan pada Bab II. Pemilihan metode Software Development Life Cycle (SDLC) yang tepat merupakan langkah strategis yang krusial untuk memastikan keberhasilan proyek serta mencegah pembengkakan biaya dan waktu pengembangan (Aniley et al., 2024). Berdasarkan pertimbangan tersebut, penelitian ini mengadopsi metode Fountain yang dinilai relevan karena memiliki karakteristik fleksibel namun tetap terstruktur. Efektivitas metode ini didukung oleh penelitian (Sastra and Sutawinata, 2023) pada perancangan sistem SIP-PTK, yang menunjukkan bahwa model Fountain mampu memandu tahapan analisis, desain, implementasi, hingga pengujian secara efektif pada sistem yang memiliki kebutuhan pengolahan data yang spesifik. Relevansi metode Fountain juga sejalan dengan fokus penelitian ini, yaitu pengembangan aplikasi *frontend* yang bersifat data-driven dan memiliki ketergantungan tinggi terhadap interaksi antar-komponen. Dalam pengembangan *frontend*, tahapan perancangan arsitektur, implementasi, dan pengujian tidak berjalan secara linier, melainkan saling berkaitan dan sering memerlukan penyesuaian berdasarkan hasil evaluasi sistem. Penerapan metode Fountain memungkinkan tahapan pengembangan dan testing berjalan secara paralel serta saling tumpang tindih (overlapping) tanpa menghilangkan struktur dan urutan penelitian yang jelas. Melalui pendekatan ini, aktivitas analisis kebutuhan, perancangan arsitektur *frontend*, implementasi *Client Data Layer*, serta pengujian sistem dapat dilakukan secara berulang dan saling mempengaruhi. Hasil pada satu tahapan dapat langsung digunakan sebagai dasar penyesuaian pada tahapan lainnya, seperti perubahan struktur data atau mekanisme sinkronisasi, tanpa harus menunggu seluruh siklus pengembangan selesai. Dengan demikian, proses pengembangan sistem menjadi lebih adaptif dan terkontrol, sehingga diharapkan mampu menghasilkan hasil penelitian yang sesuai dengan tujuan penelitian secara optimal. Tahapan penerapan metode fountain dalam penelitian ini terdiri atas beberapa tahap sebagai berikut:

### 3.2.1 Analysis

Pada tahap analisis, penelitian ini mengkaji kebutuhan dan permasalahan pada pengembangan aplikasi *frontend* berbasis web yang berfungsi sebagai dashboard analisis sentimen UMKM. Analisis dilakukan terhadap kebutuhan penyajian hasil analisis sentimen media sosial dalam bentuk visualisasi yang informatif, ringkas, dan mudah dipahami oleh pengguna. Informasi yang dianalisis mencakup kebutuhan penampilan ringkasan sentimen, distribusi sentimen, serta indikator lain yang relevan untuk memantau persepsi konsumen secara umum. Analisis juga difokuskan pada alur pengelolaan data antara backend dan *frontend*. Pada penelitian ini, proses pengumpulan data media sosial dan analisis sentimen sepenuhnya dilakukan di sisi backend, sementara *frontend* bertugas mengonsumsi data hasil analisis melalui REST API. Kondisi ini menuntut adanya mekanisme pengelolaan data *frontend* yang mampu

menerima data secara konsisten dan menyajikannya ke berbagai komponen dashboard tanpa menimbulkan inkonsistensi informasi. Selain itu, pada tahap analisis diidentifikasi permasalahan pengelolaan data *frontend* pada aplikasi dashboard yang bersifat data-driven, khususnya ketika data yang sama digunakan oleh banyak komponen antarmuka secara bersamaan. Permasalahan yang dianalisis meliputi potensi terjadinya permintaan data berulang, kesulitan sinkronisasi data antar-komponen, serta kebutuhan pembaruan data secara terkontrol. Hasil analisis ini menjadi dasar dalam menentukan kebutuhan arsitektur *frontend* yang lebih terstruktur dan efisien. Pengguna sistem dalam penelitian ini dianalisis sebagai pengguna umum atau pelaku UMKM yang memanfaatkan dashboard untuk memantau informasi sentimen. Aktivitas pengguna dibatasi pada pengamatan dan eksplorasi informasi yang ditampilkan, tanpa keterlibatan langsung dalam proses pengolahan data. Dengan karakteristik pengguna tersebut, analisis sistem difokuskan pada aspek penyajian informasi dan pengelolaan data di sisi *frontend* agar sesuai dengan tujuan penelitian.

### 3.2.2 Requirement Specification

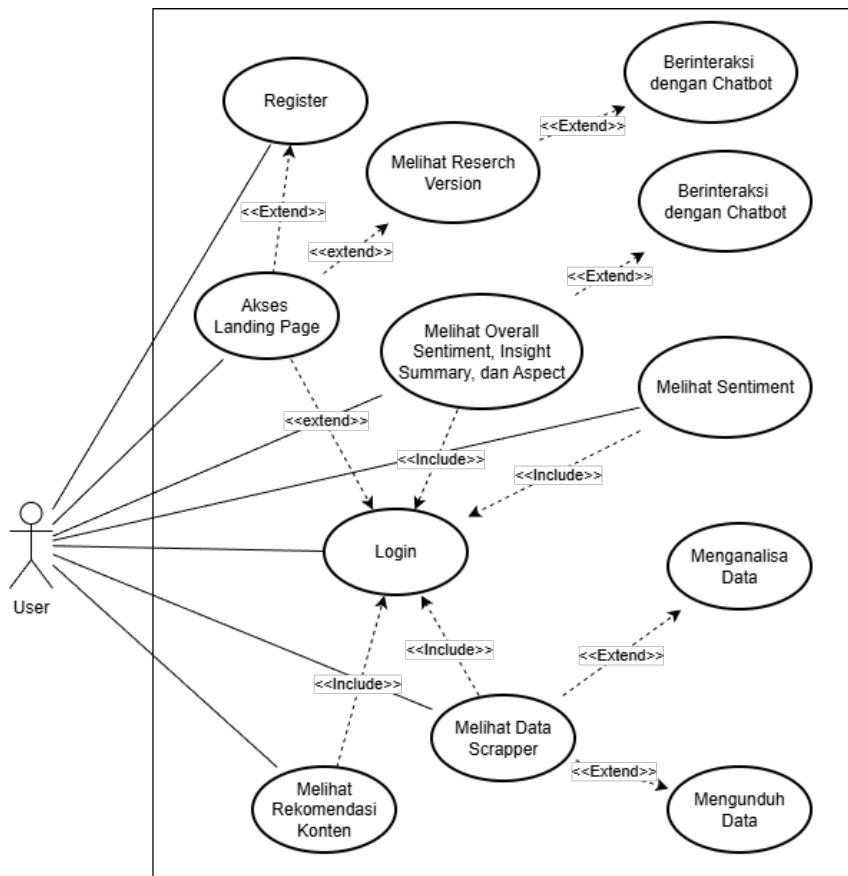
Tahap *requirement specification* bertujuan untuk merumuskan kebutuhan sistem secara terstruktur berdasarkan hasil analisis yang telah dilakukan pada tahap sebelumnya. Pada penelitian ini, spesifikasi kebutuhan difokuskan pada sisi *frontend* dashboard analisis sentimen UMKM, dengan mempertimbangkan karakteristik sistem yang bersifat data-driven dan bergantung pada data hasil analisis sentimen dari backend. Kebutuhan fungsional sistem mencakup kemampuan aplikasi *frontend* untuk mengonsumsi data hasil analisis sentimen yang disediakan melalui REST API dan menyajikannya dalam berbagai komponen dashboard. Sistem harus mampu menampilkan ringkasan sentimen, distribusi sentimen, serta indikator pendukung lainnya secara konsisten pada seluruh komponen antarmuka. Selain itu, sistem perlu mendukung pembaruan data berdasarkan hasil analisis sentimen yang dipilih pengguna, sehingga informasi yang ditampilkan tetap relevan dengan dataset terbaru tanpa memerlukan pengambilan data ulang secara manual pada setiap komponen antarmuka. Selain kebutuhan fungsional, sistem juga memiliki kebutuhan non-fungsional yang berkaitan dengan kualitas pengelolaan data dan performa aplikasi *frontend*. Kebutuhan non-fungsional tersebut meliputi konsistensi data antar-komponen antarmuka, efisiensi dalam pengambilan data dari API, serta mekanisme pengelolaan cache untuk mengurangi permintaan data yang tidak diperlukan. Sistem diharapkan mampu mengelola data secara terpusat di sisi *frontend* sehingga setiap komponen menggunakan sumber data yang sama dan tersinkronisasi. Kebutuhan lain yang menjadi perhatian pada tahap ini adalah kebutuhan kemudahan pengembangan dan pemeliharaan sistem. Struktur pengelolaan data *frontend* harus memungkinkan penambahan atau perubahan komponen dashboard tanpa

memengaruhi keseluruhan sistem secara signifikan. Dengan demikian, spesifikasi kebutuhan ini menjadi dasar dalam perancangan arsitektur *frontend* dan penerapan *Client Data Layer* pada tahap desain dan implementasi selanjutnya.

### 3.2.3 Design

Tahap *design* merupakan tahapan perancangan solusi berdasarkan spesifikasi kebutuhan sistem yang telah dirumuskan pada tahap *requirement specification*. Pada tahap ini, kebutuhan sistem yang bersifat konseptual diterjemahkan ke dalam rancangan teknis yang menjadi acuan dalam proses implementasi *frontend* dashboard analisis sentimen. Perancangan difokuskan pada bagaimana sistem *frontend* mengelola dan menyajikan data secara terstruktur, konsisten, dan mudah dikembangkan sesuai dengan tujuan penelitian. Pada penelitian ini, tahap *design* mencakup beberapa aspek utama, yaitu *use case diagram*, *entity relationship diagram*, alur sistem keseluruhan, perancangan arsitektur *frontend*, perancangan *Client Data Layer* sebagai mekanisme pengelolaan data dari API, serta perancangan antarmuka pengguna dalam bentuk *wireframe*. Setiap aspek perancangan memiliki peran yang saling berkaitan dalam membangun sistem *frontend* yang bersifat *data-driven*.

#### 1. Use Case Diagram



**Gambar 3.1** Use Case Diagram

Use Case Diagram pada Gambar 3.1 menggambarkan interaksi utama antara aktor dan sistem Dashboard Analisis Sentimen yang dikembangkan dalam penelitian ini. Aktor utama pada sistem adalah User, yang merepresentasikan pengguna akhir seperti pelaku UMKM atau pemilik bisnis yang memanfaatkan sistem untuk memperoleh informasi analisis sentimen dari data media sosial. Interaksi pengguna dengan sistem diawali melalui use case Akses Landing Page, yang dapat diakses tanpa proses autentikasi. Dari use case ini, pengguna memiliki opsi untuk melakukan Register sebagai pengguna baru atau Login bagi pengguna yang telah terdaftar, yang dimodelkan menggunakan relasi extend karena kedua proses tersebut bersifat opsional dan bergantung pada kondisi pengguna.

Setelah berhasil melakukan Login, pengguna dapat mengakses fitur-fitur utama sistem. Salah satu fitur utama adalah Melihat Sentiment, yang digunakan untuk menampilkan hasil analisis sentimen dari data yang telah diproses. Use case Melihat Sentiment memiliki relasi include dengan Menganalisa Data, yang menunjukkan bahwa proses analisis data merupakan bagian yang tidak terpisahkan dari penyajian informasi sentimen. Hasil analisis tersebut kemudian dapat diperluas melalui relasi extend ke use case Melihat Overall Sentiment, Insight Summary, dan Aspect Summary, yang memungkinkan pengguna memperoleh ringkasan sentimen secara keseluruhan, wawasan utama, serta analisis sentimen berbasis aspek.

Selain itu, sistem juga menyediakan fitur Melihat Data Scrapper yang memungkinkan pengguna melihat data hasil proses pengambilan data dari media sosial. Dari use case ini, terdapat relasi extend ke Mengunduh Data, yang menunjukkan bahwa proses pengunduhan bersifat opsional dan hanya dilakukan apabila pengguna membutuhkan data tersebut dalam bentuk berkas. Di sisi lain, pengguna juga dapat mengakses fitur Melihat Rekomendasi Konten, yang disediakan sebagai bagian dari layanan sistem untuk memberikan rekomendasi berdasarkan hasil analisis sentimen. Keseluruhan relasi use case ini menggambarkan alur penggunaan sistem secara terstruktur, mulai dari akses awal hingga pemanfaatan fitur analisis dan penyajian informasi sentimen.

## 2. Entity Relationship Diagram



**Gambar 3.2** Entity Relationship Diagram

ERD pada Gambar 3.2 menggambarkan bagaimana data hasil pengambilan data media sosial, hasil analisis sentimen, serta rekomendasi konten disimpan dan saling terhubung dalam basis data.

Entitas users berperan sebagai entitas utama yang menyimpan informasi pengguna sistem, termasuk identitas dan data autentikasi. Setiap pengguna dapat menghasilkan satu atau lebih data hasil pengambilan media sosial yang direpresentasikan oleh entitas scrape\_results, yang menyimpan informasi profil serta data mentah hasil proses scraping. Relasi antara entitas users dan scrape\_results menunjukkan hubungan satu-ke-banyak, di mana satu pengguna dapat memiliki beberapa hasil pengambilan data.

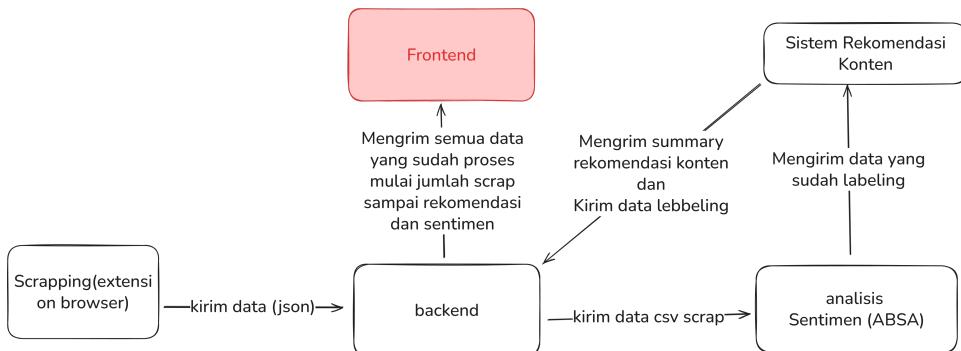
Hasil proses pengambilan data selanjutnya dianalisis dan disimpan pada entitas sentiment\_result, yang memiliki relasi satu-ke-satu atau satu-ke-banyak dengan scrape\_results tergantung pada kebutuhan analisis. Untuk mendukung analisis sentimen berbasis aspek, entitas sentiment\_comments digunakan untuk menyimpan komentar beserta kategori aspek sentimen seperti food\_quality, price, dan

service. Entitas ini memiliki relasi satu-ke-banyak dengan sentiment\_result, yang memungkinkan satu hasil analisis sentimen memiliki banyak komentar terkait.

Selain analisis sentimen, sistem juga menyediakan fitur rekomendasi konten yang dimodelkan melalui entitas recommendation\_result. Entitas ini berelasi dengan sentiment\_result sebagai dasar penyusunan rekomendasi. Detail rekomendasi disimpan pada beberapa entitas turunan, yaitu recommendation\_hashtags, recommendation\_captions, dan recommendation\_best\_posting, yang masing-masing memiliki relasi satu-ke-banyak terhadap recommendation\_result. Struktur ini memungkinkan sistem menyimpan berbagai bentuk rekomendasi secara terpisah namun tetap terhubung pada satu konteks hasil analisis.

Selain itu, entitas langchain\_documents digunakan untuk menyimpan data teks, metadata, dan embedding vektor yang mendukung proses analisis lanjutan atau pemrosesan berbasis kecerdasan buatan. Langchain\_documents berfungsi sebagai penyimpanan dokumen pendukung di luar alur utama pengelolaan data dashboard.

### 3. Alur Sistem Keseluruhan



**Gambar 3.3** Alur Sistem Keseluruhan

Alur kerja keseluruhan sistem dimulai dari Dashboard, yang berfungsi sebagai pusat interaksi utama bagi pengguna. Dari dashboard, pengguna dapat mengakses halaman Scraper untuk melakukan proses pengambilan data media sosial. Pada tahap ini, pengguna memanfaatkan layanan scraper yang terintegrasi dengan extension scraper untuk mengumpulkan data dari media sosial. Setelah proses pengambilan data selesai, data hasil scraping dikirimkan ke backend sistem dan selanjutnya disimpan ke dalam *database* sebagai data mentah.

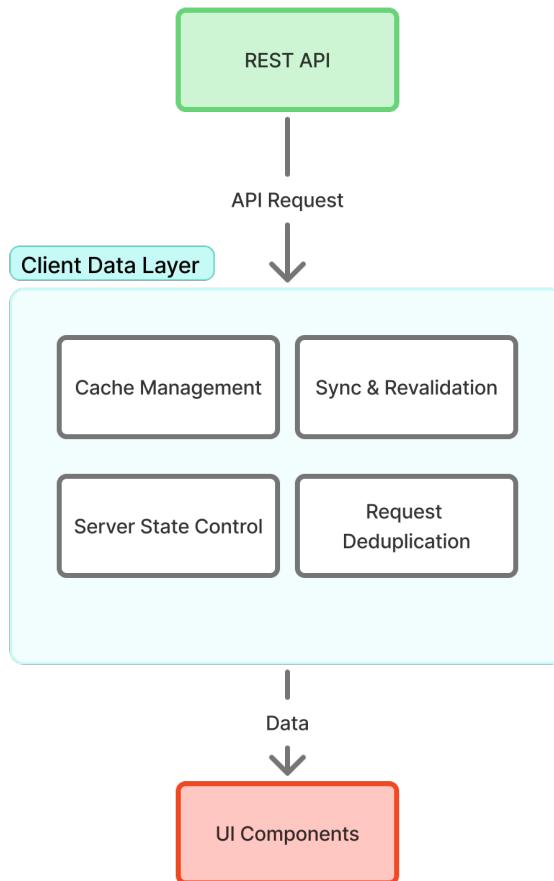
Data hasil scraping yang telah tersimpan kemudian dapat diproses lebih lanjut melalui tahap analisis data, di mana sistem melakukan analisis sentimen berbasis aspek

(Aspect-Based Sentiment Analysis/ABSA). Proses ABSA ini menghasilkan informasi sentimen yang terstruktur berdasarkan aspek-aspek tertentu. Hasil analisis sentimen tersebut selanjutnya diteruskan ke modul rekomendasi konten, yang berfungsi untuk menghasilkan rekomendasi konten berdasarkan pola dan hasil sentimen yang diperoleh.

Setelah proses rekomendasi selesai, data rekomendasi konten beserta hasil analisis sentimen dikirim kembali ke backend dan disimpan ke dalam *database*. Pada tahap akhir, sistem menyajikan kembali data hasil scraping, hasil analisis sentimen, serta rekomendasi konten ke Dashboard, sehingga pengguna dapat melihat, mengevaluasi, dan memanfaatkan seluruh informasi yang dihasilkan oleh sistem secara terintegrasi.

#### 4. Perancangan Arsitektur *Frontend*

Penelitian ini menerapkan prinsip *component-based architecture*, di mana antarmuka pengguna dibangun dari komponen-komponen modular yang memiliki tanggung jawab spesifik dan dapat digunakan kembali. Setiap komponen difokuskan pada penyajian data dan interaksi pengguna, sementara logika pengelolaan data dipisahkan ke dalam lapisan tersendiri agar struktur aplikasi lebih terorganisasi dan mudah dipelihara. Arsitektur *frontend* dirancang dengan pendekatan *data-driven*, di mana tampilan antarmuka sepenuhnya bergantung pada data yang dikelola oleh sistem. Untuk mendukung hal tersebut, prinsip *separation of concerns* diterapkan dengan memisahkan lapisan presentasi dan lapisan pengelolaan data, sehingga komponen antarmuka tidak berinteraksi langsung dengan REST API.



**Gambar 3.4** Diagram Arsitektur *Frontend*

Diagram arsitektur *frontend* pada Gambar 3.4 digunakan untuk menggambarkan pembagian lapisan sistem serta alur pengelolaan data pada sistem yang dikembangkan. Diagram ini menunjukkan bagaimana data dari REST API dikelola melalui *Client Data Layer* sebelum disajikan pada komponen antarmuka pengguna. Lapisan REST API diposisikan sebagai sumber data eksternal yang menyediakan data hasil analisis sentimen. Seluruh proses pengolahan data, termasuk pengambilan data media sosial dan analisis sentimen, dilakukan pada sisi backend dan berada di luar ruang lingkup penelitian ini. *Frontend* berperan sebagai konsumen data yang mengakses informasi tersebut melalui antarmuka REST API. Lapisan *Client Data Layer* berfungsi sebagai lapisan perantara antara REST API dan komponen antarmuka pengguna. Data yang diperoleh dari REST API dikelola dan dimodelkan secara terpusat sebelum disajikan pada komponen antarmuka pengguna. Lapisan ini bertanggung jawab dalam proses pengambilan data, penyimpanan sementara (*caching*), serta sinkronisasi data antar-komponen. Lapisan *UI Components* merupakan lapisan presentasi yang bertugas menampilkan data kepada pengguna dan menangani interaksi pengguna dengan sistem. Komponen pada lapisan ini menerima data yang telah dikelola oleh

*Client Data Layer* dan menyajikannya dalam bentuk visualisasi seperti grafik dan tabel.

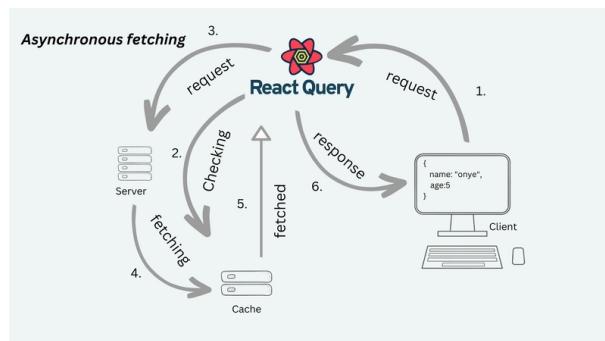
## 5. Perancangan *Client Data Layer*

Perancangan *Client Data Layer* dilakukan untuk mengelola data yang bersumber dari backend secara terpusat pada sisi *frontend* sebelum disajikan pada komponen antarmuka pengguna. Pada aplikasi dashboard yang bersifat data-driven, data yang sama dapat digunakan oleh berbagai komponen secara bersamaan dan diperbarui secara dinamis. Oleh karena itu, diperlukan suatu lapisan pengelolaan data yang mampu mengatur alur data, menjaga konsistensi informasi, serta mengendalikan interaksi antara *frontend* dan REST API. *Client Data Layer* diposisikan sebagai lapisan perantara antara REST API dan komponen antarmuka pengguna, sebagaimana ditunjukkan pada Gambar 3.4 diagram arsitektur *frontend*. Seluruh data yang diperoleh dari backend tidak langsung digunakan oleh komponen antarmuka, melainkan terlebih dahulu dikelola melalui *Client Data Layer*. Dengan pendekatan ini, komponen antarmuka tidak perlu mengetahui detail proses pengambilan data dari API, sehingga fokus komponen dapat diarahkan pada penyajian data dan interaksi pengguna. Secara konseptual, *Client Data Layer* memiliki beberapa tanggung jawab utama dalam sistem *frontend*. Tanggung jawab tersebut meliputi proses pengambilan data dari REST API, pengelolaan server state, penyimpanan sementara data melalui mekanisme caching, serta sinkronisasi data antar-komponen antarmuka. Dengan pengelolaan data yang terpusat, permintaan data yang bersifat berulang dapat dikendalikan dan setiap komponen antarmuka memperoleh data yang konsisten sesuai dengan kondisi sistem. Selain pengelolaan alur data, *Client Data Layer* juga dirancang untuk menangani pemodelan data sebelum digunakan oleh komponen antarmuka. Data yang diperoleh dari REST API dimodelkan secara terstruktur pada *Client Data Layer* agar memiliki bentuk dan konsistensi yang jelas. Pendekatan ini bertujuan untuk meminimalkan ketergantungan komponen antarmuka terhadap struktur data mentah dari backend serta mempermudah proses pengembangan dan pemeliharaan sistem *frontend*. Dalam penelitian ini, *Client Data Layer* dirancang untuk diimplementasikan menggunakan TanStack Query sebagai pustaka pengelolaan server state pada *frontend*. Pemilihan TanStack Query didasarkan pada kemampuannya dalam menyediakan mekanisme pengelolaan data asinkron secara terpusat, termasuk caching, sinkronisasi data, dan pengendalian permintaan data. Dengan memanfaatkan pustaka tersebut, *Client Data Layer* diharapkan mampu

mendukung pengelolaan data *frontend* yang lebih terstruktur, konsisten, dan efisien sesuai dengan kebutuhan dashboard analisis sentimen.

## 6. Perancangan Penggunaan TanStack Query

Perancangan penggunaan TanStack Query pada penelitian ini mengacu pada dokumentasi resmi TanStack Query sebagai pustaka server state management untuk aplikasi *frontend*. Berdasarkan dokumentasi resmi TanStack Query (Tanstack LCC, 2025), pustaka ini dirancang untuk mengelola data asinkron yang bersumber dari API secara terpusat melalui mekanisme pengambilan data, penyimpanan sementara (caching), serta sinkronisasi data antar-komponen antarmuka. Pendekatan tersebut memungkinkan komponen *frontend* memperoleh data yang konsisten tanpa harus melakukan permintaan data secara langsung ke REST API, sehingga pemisahan tanggung jawab antara lapisan presentasi dan lapisan pengelolaan data dapat terjaga. Dengan karakteristik tersebut, TanStack Query dinilai sesuai untuk diimplementasikan sebagai *Client Data Layer* pada aplikasi *frontend* yang bersifat data-driven, khususnya dalam konteks dashboard analisis sentimen yang membutuhkan konsistensi data dan pembaruan informasi secara terkontrol.



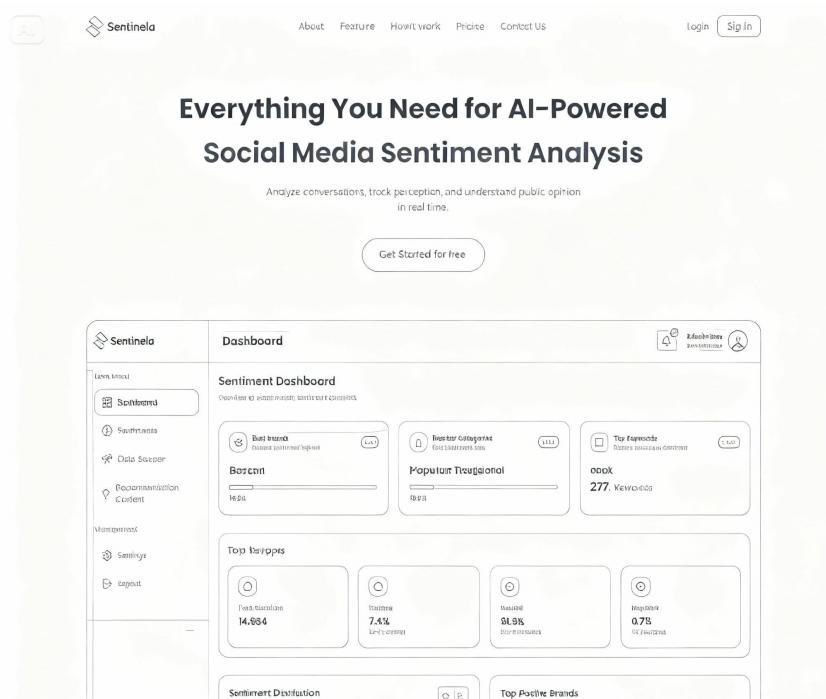
**Gambar 3.5** Cara Kerja TanStack Query

Gambar 3.5 menunjukkan alur pengelolaan data asinkron pada sisi *frontend* menggunakan TanStack Query. Ketika komponen antarmuka membutuhkan data, permintaan tidak langsung dikirimkan ke REST API, melainkan terlebih dahulu dikelola oleh *Client Data Layer*. TanStack Query melakukan pengecekan terhadap cache untuk menentukan apakah data yang diminta masih valid. Jika data tersedia dan masih relevan, data langsung dikembalikan ke komponen antarmuka tanpa melakukan pemanggilan ulang ke server. Sebaliknya, apabila data tidak tersedia atau sudah tidak valid, sistem akan melakukan proses fetching ke REST API dan menyimpan hasilnya ke dalam cache sebelum disajikan ke antarmuka pengguna. Mekanisme ini

memungkinkan pengelolaan data yang lebih efisien, mengurangi jumlah permintaan API yang tidak perlu, serta menjaga konsistensi data antar-komponen pada *frontend* dashboard analisis sentimen.

## 7. Perancangan Antarmuka

Rancangan antarmuka dilakukan sebagai tindak lanjut dari struktur arsitektur yang telah ditetapkan, dengan tujuan memastikan bahwa data hasil analisis sentimen yang dikelola oleh sistem dapat disajikan kepada pengguna secara informatif, mudah dipahami, dan konsisten. Antarmuka pengguna dirancang untuk merepresentasikan kebutuhan fungsional sistem dalam bentuk visual, sekaligus menjadi media interaksi antara pengguna dan sistem *frontend* dashboard analisis sentimen. Sebagai bentuk konkret dari perancangan antarmuka pengguna, dilakukan penyusunan wireframe yang menggambarkan tata letak komponen, alur navigasi, serta penyajian informasi pada setiap halaman utama sistem.



Gambar 3.6 Wireframe Landing Page

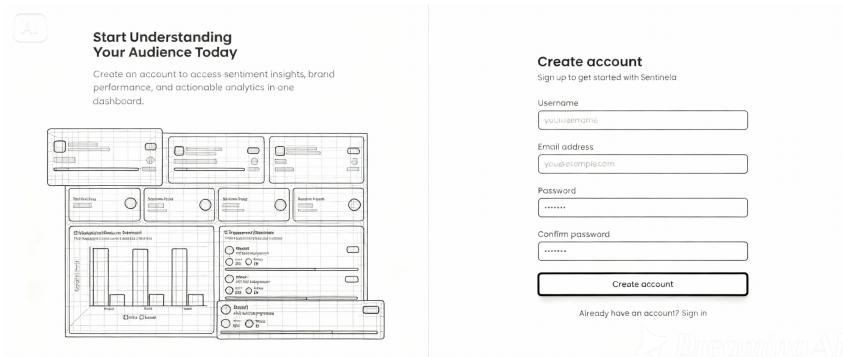
Landing Page pada Gambar 3.6 dirancang sebagai halaman awal yang pertama kali diakses oleh pengguna ketika membuka sistem. Halaman ini berfungsi untuk memberikan gambaran umum mengenai tujuan dan fitur utama sistem sebelum pengguna melakukan proses autentikasi. Struktur halaman dirancang sederhana dengan penekanan pada informasi pengenalan sistem serta elemen navigasi utama yang mengarahkan pengguna ke halaman login atau registrasi. Perancangan

wireframe ini bertujuan untuk memastikan pengguna dapat memahami konteks sistem secara cepat dan memiliki alur navigasi yang jelas menuju fitur utama yang disediakan.



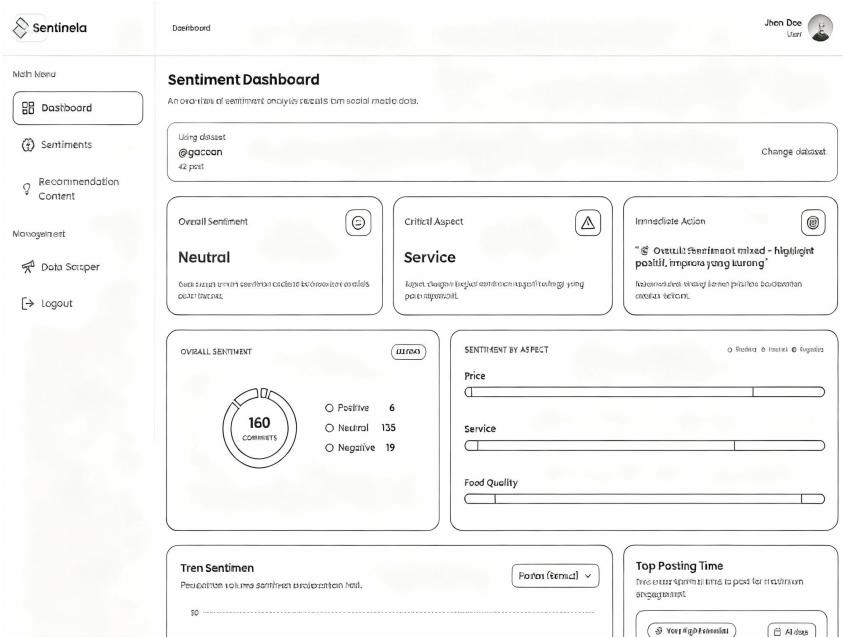
**Gambar 3.7** Wireframe Halaman Login

Halaman Login pada Gambar 3.7 dirancang sebagai antarmuka autentikasi pengguna untuk mengakses fitur utama sistem. Halaman ini menyediakan elemen input untuk memasukkan kredensial pengguna untuk memproses autentikasi. Pengguna dapat melakukan proses login sebelum diarahkan ke halaman dashboard. Halaman ini dirancang untuk mendukung keamanan akses sistem dengan memastikan bahwa hanya pengguna yang telah terautentikasi yang dapat mengakses fitur-fitur utama aplikasi.



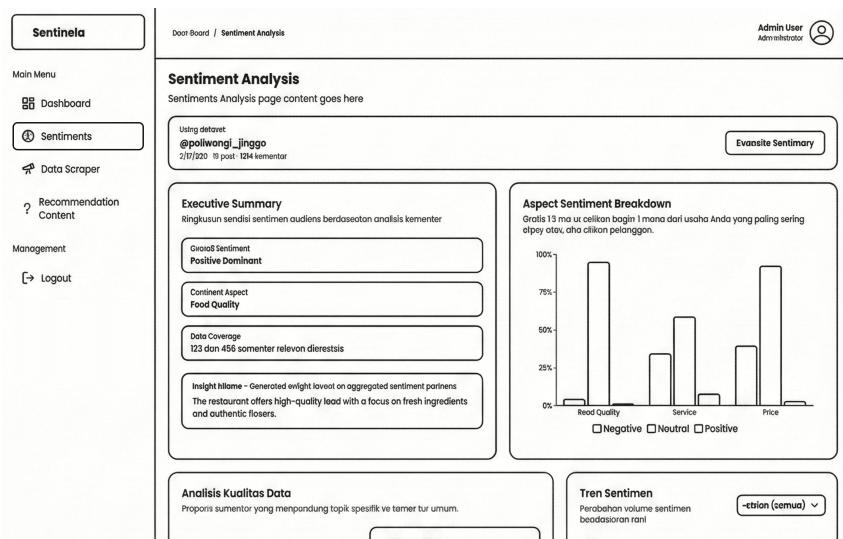
**Gambar 3.8** Wireframe Halaman Register

Halaman Register pada Gambar 3.8 dirancang sebagai antarmuka pendaftaran pengguna baru sebelum dapat mengakses sistem. Halaman ini menyediakan form untuk pengisian data pengguna yang diperlukan dalam proses registrasi. Pengguna dapat melakukan proses pendaftaran secara sistematis. Halaman ini juga berfungsi sebagai bagian dari mekanisme kontrol akses dengan memastikan bahwa data pengguna dikumpulkan dan diproses sebelum akun dapat digunakan untuk mengakses fitur utama sistem.



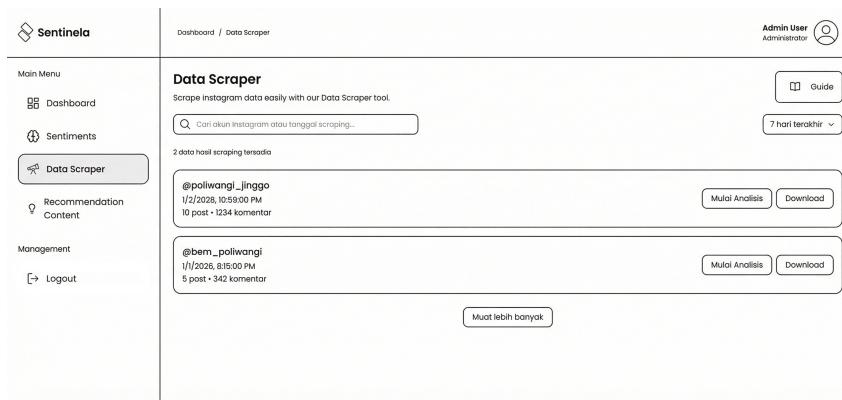
**Gambar 3.9** Wireframe Halaman Dashboard

Halaman Dashboard pada Gambar 3.9 dirancang sebagai halaman utama setelah pengguna berhasil melakukan autentikasi. Halaman ini berfungsi sebagai pusat informasi yang menampilkan ringkasan data dan visualisasi utama dari sistem. Struktur dashboard dirancang untuk memudahkan pengguna dalam memantau kondisi data secara keseluruhan serta mengakses fitur-fitur utama melalui navigasi yang tersedia. Perancangan wireframe ini bertujuan untuk memastikan penyajian informasi yang terstruktur dan mudah dipahami, sehingga pengguna dapat memperoleh gambaran umum hasil analisis secara cepat sebelum melakukan eksplorasi data lebih lanjut.



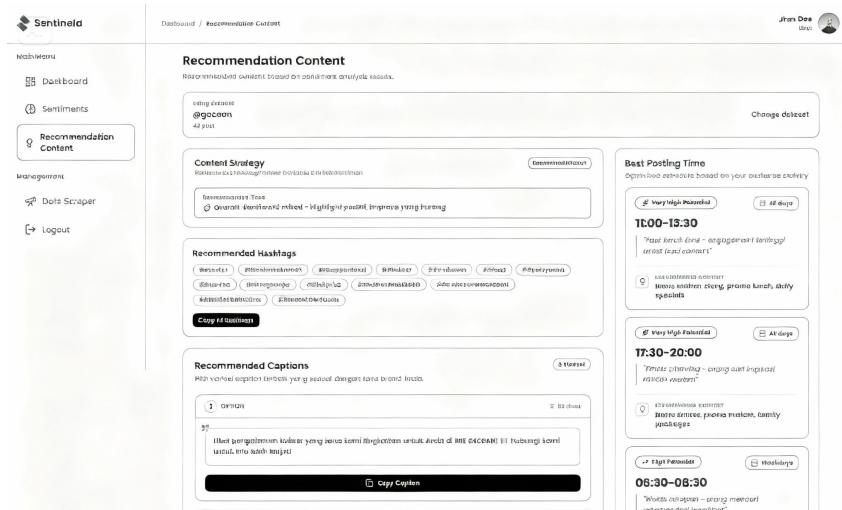
**Gambar 3.10** Wireframe Halaman Sentiment

Halaman Sentiment pada Gambar 3.10 dirancang untuk menampilkan hasil analisis sentimen aspect based sentiment analysis secara lebih rinci dibandingkan halaman dashboard. Halaman ini menyajikan informasi sentimen dalam bentuk visualisasi data yang memudahkan pengguna dalam memahami distribusi dan kecenderungan sentimen. Perancangan struktur halaman difokuskan pada penyajian data yang terorganisasi dan mudah diinterpretasikan, sehingga pengguna dapat melakukan analisis sentimen secara lebih mendalam sesuai dengan kebutuhan informasi yang diinginkan.



**Gambar 3.11** Wireframe Halaman Scraper

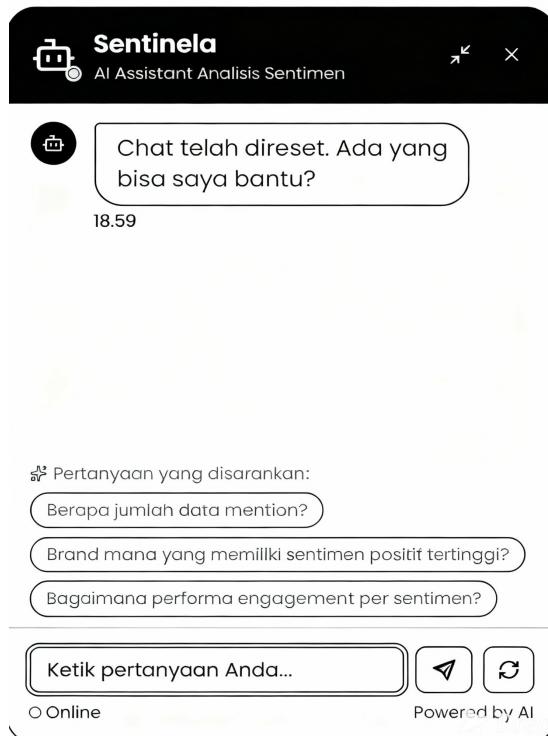
Halaman Scraper pada Gambar 3.11 dirancang sebagai antarmuka yang memfasilitasi proses pengumpulan dan pemantauan data yang bersumber dari media sosial. Halaman ini menyajikan data hasil proses scraping yang dilakukan pada sisi server, sehingga pengguna dapat mengetahui data yang tersedia dan data yang mana yang akan digunakan oleh sistem pada tahap analisis.



**Gambar 3.12** Wireframe Halaman Recomendation

Halaman Recomendation pada Gambar 3.12 dirancang sebagai antarmuka yang

menyajikan rekomendasi berdasarkan hasil analisis sentimen yang telah diproses secara otomatis oleh sistem. Halaman ini menampilkan informasi rekomendasi yang diperoleh dari pengolahan data sentimen, sehingga pengguna dapat memahami insight yang dihasilkan dari data media sosial. Hasil rekomendasi yang ditampilkan pada halaman ini juga disajikan secara lebih detail sebagai bagian dari informasi rekomendasi utama. Halaman ini bertujuan untuk memudahkan pengguna dalam mengakses rekomendasi dan memahami hasil analisis sentimen secara terstruktur.



**Gambar 3.13** Wireframe Halaman Chatbot

Halaman Chatbot pada Gambar 3.13 dirancang sebagai antarmuka yang memfasilitasi interaksi tanya jawab antara pengguna dengan sistem. Halaman ini memungkinkan pengguna untuk mengajukan pertanyaan terkait data yang tersedia dan memperoleh jawaban yang relevan berdasarkan informasi yang telah diproses oleh sistem. Halaman ini bertujuan untuk memudahkan pengguna dalam mengakses informasi dan memperoleh insight yang dibutuhkan secara interaktif.

#### 3.2.4 Coding (Implementation)

Tahap coding diawali dengan penyiapan lingkungan pengembangan *frontend* dan penyesuaian struktur proyek agar selaras dengan arsitektur yang telah dirancang. Framework React dipilih sebagai fondasi antarmuka pengguna, sedangkan TanStack Query diterapkan sebagai mekanisme utama pengelolaan server state yang membentuk *Client Data Layer*.

Dalam mengimplementasikan struktur direktori dan manajemen pengambilan data (data *fetching*), penelitian ini mengadopsi pendekatan arsitektur berlapis (*layered architecture*) yang direkomendasikan dalam praktik produksi React modern (Owusu, 2025). Sesuai pola tersebut, implementasi dilakukan dengan memisahkan logika API ke dalam service layer, yang kemudian dibungkus menggunakan custom hooks berbasis TanStack Query. Pendekatan ini memastikan bahwa komponen UI hanya berinteraksi dengan hooks tanpa perlu mengetahui kompleksitas implementasi REST API secara langsung.

Bersamaan dengan itu, dilakukan pula persiapan komponen pendukung dan integrasi data dengan backend. Seluruh persiapan ini bertujuan untuk menjamin bahwa proses implementasi berjalan secara terstruktur, menjaga konsistensi kode, serta memenuhi spesifikasi kebutuhan sistem yang telah didefinisikan pada tahap sebelumnya.

### 3.2.5 Testing

Metode pengujian yang digunakan dalam penelitian ini mengacu pada konsep black-box testing, dengan pendekatan scenario-based testing. Pengujian dilakukan dengan mengamati perilaku sistem berdasarkan skenario penggunaan tanpa memperhatikan struktur internal kode program. Pendekatan ini menempatkan sistem sebagai sebuah kesatuan yang diuji dari sudut pandang pengguna, sehingga pengujian difokuskan pada kesesuaian fungsi dan respons sistem terhadap alur penggunaan yang dirancang. Pemilihan pendekatan scenario-based testing didasarkan pada karakteristik sistem yang dikembangkan, yaitu dashboard analitik yang bersifat data-driven dan mengandalkan interaksi antar-komponen antarmuka. Oleh karena itu, pengujian diarahkan pada pengetesan perilaku sistem dalam menampilkan data, menjaga konsistensi informasi, serta merespons pembaruan data sesuai dengan kondisi yang terjadi. Ruang lingkup pengujian pada penelitian ini difokuskan pada perilaku sistem *frontend*, khususnya pada pengelolaan data melalui *Client Data Layer* dan penyajian data pada antarmuka pengguna. Pengujian tidak mencakup pengetesan terhadap algoritma analisis sentimen maupun proses pengolahan data pada sisi backend. Skenario pengujian disusun berdasarkan fitur utama sistem dan merepresentasikan alur penggunaan dari sudut pandang pengguna. Setiap skenario dirancang untuk mengevaluasi perilaku sistem *frontend* dalam merespons interaksi pengguna serta memastikan kesesuaian fungsi sistem dengan rancangan yang telah ditetapkan. Skenario pengujian dalam penelitian ini terdiri atas beberapa pengujian sebagai berikut:

1. Landing Page

**Tabel 3.2** Skenario Pengujian Landing Page

ID	Test Step	Skenario Pengujian	Data Uji	Expected Result
TC-LP-01	<ul style="list-style-type: none"> <li>(a) Pengguna membuka browser</li> <li>(b) Pengguna mengakses URL aplikasi</li> <li>(c) Sistem memuat halaman awal</li> </ul>	Pengguna mengakses aplikasi tanpa melakukan autentikasi	–	Halaman landing ditampilkan dengan informasi sistem serta navigasi menuju halaman login dan registrasi.

2. Login

**Tabel 3.3** Skenario Pengujian Halaman Login

ID	Test Step	Skenario Pengujian	Data Uji	Hasil yang Diharapkan
TC-LG-01	<ul style="list-style-type: none"> <li>(a) Pengguna membuka halaman login</li> <li>(b) Pengguna mengisi username dan kata sandi valid</li> <li>(c) Pengguna menekan tombol login</li> </ul>	Login dengan kredensial yang valid	username dan kata sandi valid	Sistem menerima kredensial dan mengarahkan pengguna ke halaman dashboard.

<b>ID</b>	<b>Test Step</b>	<b>Skenario Pengujian</b>	<b>Data Uji</b>	<b>Hasil yang Diharapkan</b>
TC-LG-02	<ul style="list-style-type: none"> <li>(a) Pengguna membuka halaman login</li> <li>(b) Pengguna mengisi username valid</li> <li>(c) Pengguna mengisi kata sandi salah</li> <li>(d) Pengguna menekan tombol login</li> </ul>	Login dengan kata sandi yang salah	username valid, kata sandi salah	Sistem menampilkan pesan kesalahan dan tetap berada di halaman login.
TC-LG-03	<ul style="list-style-type: none"> <li>(a) Pengguna membuka halaman login</li> <li>(b) Pengguna mengisi username yang tidak terdaftar</li> <li>(c) Pengguna mengisi kata sandi</li> <li>(d) Pengguna menekan tombol login</li> </ul>	Login dengan username yang tidak terdaftar	username tidak terdaftar	Sistem menampilkan pesan bahwa akun tidak ditemukan.
TC-LG-04	<ul style="list-style-type: none"> <li>(a) Pengguna membuka halaman login</li> <li>(b) Pengguna tidak mengisi username atau kata sandi</li> <li>(c) Pengguna menekan tombol login</li> </ul>	Login dengan field kosong	username atau kata sandi kosong	Sistem menampilkan validasi bahwa seluruh field wajib diisi.

ID	Test Step	Skenario Pengujian	Data Uji	Hasil yang Diharapkan
TC-LG-05	<ul style="list-style-type: none"> <li>(a) Pengguna membuka halaman login</li> <li>(b) Pengguna mengisi ulang kredensial yang benar</li> <li>(c) Pengguna menekan tombol login</li> </ul>	Login berhasil setelah percobaan gagal	Kredensial valid	Sistem menerima kredensial dan mengarahkan pengguna ke halaman dashboard.

### 3. Register

**Tabel 3.4** Skenario Pengujian Halaman Register

ID	Test Step	Skenario Pengujian	Data Uji	Hasil yang Diharapkan
TC-RG-01	<ul style="list-style-type: none"> <li>(a) Pengguna membuka halaman register</li> <li>(b) Pengguna mengisi seluruh field registrasi dengan data valid</li> <li>(c) Pengguna menekan tombol daftar</li> </ul>	Registrasi dengan data yang valid	Data registrasi lengkap dan valid	Sistem menyimpan data pengguna dan akun dapat digunakan untuk login.
TC-RG-02	<ul style="list-style-type: none"> <li>(a) Pengguna membuka halaman register</li> <li>(b) Pengguna tidak mengisi salah satu atau beberapa field</li> <li>(c) Pengguna menekan tombol daftar</li> </ul>	Registrasi dengan data tidak lengkap	Salah satu atau beberapa field kosong	Sistem menampilkan pesan validasi bahwa seluruh data registrasi wajib diisi.

ID	Test Step	Skenario Pengujian	Data Uji	Hasil yang Diharapkan
TC-RG-03	(a) Pengguna membuka halaman register (b) Pengguna mengisi username yang sudah terdaftar (c) Pengguna menekan tombol daftar	Registrasi dengan username yang sudah terdaftar	username sudah terdaftar	Sistem menampilkan pesan bahwa akun sudah terdaftar.
TC-RG-04	(a) Pengguna membuka halaman register (b) Pengguna mengisi ulang data registrasi dengan benar (c) Pengguna menekan tombol daftar	Registrasi berhasil setelah kesalahan sebelumnya	Data registrasi valid	Sistem menerima data dan memproses registrasi akun.

#### 4. Dashboard

**Tabel 3.5** Skenario Pengujian Halaman Dashboard

ID	Test Step	Skenario Pengujian	Data Uji	Hasil yang Diharapkan
TC-DB-01	(a) Pengguna berhasil login (b) Sistem mengarahkan pengguna ke halaman dashboard	Membuka dashboard setelah login	Data dashboard tersedia	Sistem menampilkan ringkasan data dan visualisasi utama pada dashboard.

<b>ID</b>	<b>Test Step</b>	<b>Skenario Pengujian</b>	<b>Data Uji</b>	<b>Hasil yang Diharapkan</b>
TC-DB-02	(a) Pengguna login ke sistem (b) Sistem memuat halaman dashboard (c) Data dashboard belum tersedia	Membuka dashboard dengan data belum tersedia	Data belum tersedia	Sistem menampilkan indikator pemuatan atau pesan informasi kepada pengguna.
TC-DB-03	(a) Pengguna membuka dashboard (b) Pengguna berpindah ke menu lain (c) Pengguna kembali ke halaman dashboard	Navigasi keluar dan kembali ke dashboard	-	Sistem menampilkan data dashboard secara konsisten tanpa kehilangan data.
TC-DB-04	(a) Pengguna berada di halaman dashboard (b) Sistem menerima pembaruan data dari server	Pembaruan data dashboard dari server	Data diperbarui	Sistem memperbarui tampilan dashboard sesuai dengan data terbaru.
TC-DB-05	(a) Pengguna membuka halaman dashboard (b) Beberapa komponen memuat data dari sumber yang sama	Konsistensi data antar komponen dashboard	Data yang sama digunakan	Sistem menampilkan data yang konsisten pada seluruh komponen dashboard.

## 5. Sentiment

**Tabel 3.6** Skenario Pengujian Halaman Dashboard Sentiment

ID	Test Step	Skenario Pengujian	Data Uji	Hasil yang Diharapkan
TC-ST-01	(a) Pengguna berhasil login (b) Pengguna membuka halaman dashboard sentiment	Membuka halaman dashboard sentiment	Data sentimen tersedia	Sistem menampilkan visualisasi dan ringkasan hasil analisis sentimen.
TC-ST-02	(a) Pengguna membuka halaman dashboard sentiment (b) Data sentimen belum tersedia	Membuka dashboard sentiment dengan data belum tersedia	Data sentimen belum tersedia	Sistem menampilkan indikator pemuatan atau pesan informasi kepada pengguna.
TC-ST-03	(a) Pengguna berada di halaman dashboard sentiment (b) Sistem menerima pembaruan data sentimen dari server	Pembaruan data sentimen dari server	Data sentimen diperbarui	Sistem memperbarui visualisasi sesuai dengan data sentimen terbaru.
TC-ST-04	(a) Pengguna membuka dashboard sentiment (b) Pengguna berpindah ke halaman lain (c) Pengguna kembali ke dashboard sentiment	Navigasi keluar dan kembali ke dashboard sentiment	–	Sistem menampilkan data sentimen secara konsisten tanpa kehilangan data.

ID	Test Step	Skenario Pengujian	Data Uji	Hasil yang Diharapkan
TC-ST-05	<ul style="list-style-type: none"> <li>(a) Pengguna membuka dashboard sentiment</li> <li>(b) Beberapa komponen memuat data sentimen yang sama</li> </ul>	Konsistensi data sentimen antar komponen	Data sentimen yang sama digunakan	Sistem menampilkan data yang konsisten pada seluruh komponen visualisasi.

## 6. Recommendation

**Tabel 3.7 Skenario Pengujian Halaman Dashboard Rekomendasi Konten**

ID	Fitur	Skenario Pengujian	Data Uji	Hasil yang Diharapkan
TC-RC-01	Dashboard Rekomendasi	Pengguna membuka halaman dashboard rekomendasi konten	Data rekomendasi tersedia	Sistem menampilkan daftar rekomendasi berdasarkan hasil analisis sentimen
TC-RC-02	Dashboard Rekomendasi	Pengguna membuka halaman rekomendasi dengan data belum tersedia	Data rekomendasi belum tersedia	Sistem menampilkan indikator pemuatan atau pesan informasi

ID	Fitur	Skenario Pengujian	Data Uji	Hasil yang Diharapkan
TC-RC-03	Dashboard Rekomendasi	Pembaruan data rekomendasi dari server	Data rekomendasi diperbarui	Sistem memperbarui tampilan rekomendasi sesuai data terbaru
TC-RC-04	Dashboard Rekomendasi	Navigasi ke halaman lain dan kembali ke dashboard rekomendasi	–	Sistem menampilkan data rekomendasi secara konsisten
TC-RC-05	Dashboard Rekomendasi	Konsistensi rekomendasi dengan data sentimen	Data sentimen terkait berubah	Rekomendasi konten menyesuaikan perubahan data sentimen

## 7. Scraper

**Tabel 3.8** Skenario Pengujian Halaman Data Scraper

ID	Test Step	Skenario Pengujian	Data Uji	Hasil yang Diharapkan
TC-SC-01	(a) Pengguna membuka halaman data scraper (b) Sistem memuat data hasil scraping	Membuka halaman data scraper	Data hasil scraping tersedia	Sistem menampilkan daftar data hasil scraping yang diperoleh dari server.

<b>ID</b>	<b>Test Step</b>	<b>Skenario Pengujian</b>	<b>Data Uji</b>	<b>Hasil yang Diharapkan</b>
TC-SC-02	(a) Pengguna membuka halaman data scraper (b) Data hasil scraping belum tersedia	Membuka halaman scraper dengan data belum tersedia	Data belum tersedia	Sistem menampilkan indikator pemuatan atau pesan informasi kepada pengguna.
TC-SC-03	(a) Pengguna membuka halaman data scraper (b) Pengguna memilih salah satu dataset	Pemilihan data untuk dianalisis	Dataset tertentu dipilih	Sistem menandai data terpilih untuk digunakan pada proses analisis.
TC-SC-04	(a) Pengguna membuka halaman data scraper (b) Pengguna berpindah ke halaman lain (c) Pengguna kembali ke halaman data scraper	Navigasi keluar dan kembali ke halaman data scraper	–	Sistem menampilkan data scraper secara konsisten tanpa kehilangan data.

## 8. Chatbot

**Tabel 3.9** Skenario Pengujian Chatbot

ID	Test Step	Skenario Pengujian	Data Uji	Hasil yang Diharapkan
TC-CB-01	(a) Pengguna berhasil login (b) Pengguna berada di halaman dashboard (c) Pengguna membuka panel chatbot yang bersifat <i>sticky</i>	Membuka antarmuka chatbot pada dashboard	–	Sistem menampilkan antarmuka chatbot dan siap menerima input pengguna.
TC-CB-02	(a) Pengguna membuka chatbot pada dashboard (b) Pengguna memasukkan pertanyaan teks (c) Pengguna mengirimkan pertanyaan	Mengirimkan pertanyaan melalui chatbot	Pertanyaan teks valid	Sistem menampilkan respons chatbot sesuai dengan pertanyaan yang diberikan.
TC-CB-03	(a) Pengguna membuka chatbot (b) Pengguna mengirimkan input kosong	Mengirimkan input kosong pada chatbot	Input kosong	Sistem menampilkan pesan validasi atau informasi bahwa input tidak valid.
TC-CB-04	(a) Pengguna membuka chatbot (b) Pengguna mengirimkan pertanyaan di luar konteks sistem	Mengirimkan pertanyaan di luar konteks sistem	Pertanyaan tidak relevan	Sistem menampilkan respons atau pesan informasi yang sesuai.

ID	Test Step	Skenario Pengujian	Data Uji	Hasil yang Diharapkan
TC-CB-05	<ul style="list-style-type: none"> <li>(a) Pengguna membuka chatbot</li> <li>(b) Pengguna melakukan beberapa interaksi berturut-turut</li> </ul>	Interaksi berulang dengan chatbot	Beberapa pertanyaan berurutan	<p>Sistem memproses interaksi chatbot secara berurutan dengan menonaktifkan input selama proses berlangsung dan mengaktifkannya kembali setelah respons ditampilkan..</p>

## 9. Logout

**Tabel 3.10** Skenario Pengujian Logout

ID	Test Step	Skenario Pengujian	Data Uji	Hasil yang Diharapkan
TC-LO-01	<ul style="list-style-type: none"> <li>(a) Pengguna berada dalam kondisi login</li> <li>(b) Pengguna menekan tombol logout</li> </ul>	Melakukan logout dari sistem	–	<p>Sistem mengakhiri sesi pengguna dan mengarahkan ke halaman login.</p>

<b>ID</b>	<b>Test Step</b>	<b>Skenario Pengujian</b>	<b>Data Uji</b>	<b>Hasil yang Diharapkan</b>
TC-LO-02	(a) Pengguna telah melakukan logout (b) Pengguna mencoba mengakses halaman dashboard	Mengakses dashboard setelah logout	–	Sistem menolak akses dan mengarahkan pengguna ke halaman login.
TC-LO-03	(a) Pengguna menutup sesi aplikasi (b) Pengguna membuka kembali aplikasi	Membuka ulang aplikasi setelah logout	–	Sistem meminta pengguna untuk melakukan login kembali.

*– Halaman ini sengaja dikosongkan –*

## **BAB 4**

### **HASIL DAN PEMBAHASAN**

#### **4.1 Hasil Jawa Timur Datathon 2025**

##### **4.1.1 Capaian Jawa Timur Datathon 2025 Skema UMKM**

Jawa Timur Datathon 2025 merupakan kompetisi analisis data yang diselenggarakan oleh Dinas Komunikasi dan Informatika Provinsi Jawa Timur dengan tema “*Shaping The Future with Open Data*”, yang bertujuan mendorong pemanfaatan Open Data Jawa Timur sebagai dasar dalam analisis data dan perumusan solusi berbasis data. Keikutsertaan dalam kompetisi ini menghasilkan capaian awal berupa perumusan ide, penyusunan proposal, serta pengembangan prototipe solusi yang berangkat dari permasalahan nyata dan data terbuka yang tersedia.

Capaian yang diperoleh pada Jawa Timur Datathon 2025 menjadi dasar pengembangan penelitian pada Tugas Akhir ini. Ide solusi, rancangan awal sistem, serta pendekatan analisis yang dikembangkan selama kompetisi tidak berhenti pada konteks lomba, melainkan dilanjutkan dan disempurnakan melalui proses penelitian yang terstruktur. Pada tahap lanjutan, sistem dikembangkan kembali dengan memperhatikan aspek arsitektur *frontend*, penerapan arsitektur *Client Data Layer* menggunakan TanStack Query, serta evaluasi mekanisme pengelolaan data *frontend* sesuai dengan metodologi penelitian yang digunakan.

Dengan demikian, capaian Jawa Timur Datathon 2025 tidak hanya dipandang sebagai prestasi kompetitif, tetapi juga sebagai kontribusi awal yang mendukung pengembangan sistem dan penelitian secara akademik. Integrasi antara hasil lomba dan pengembangan lanjutan memungkinkan penelitian ini memiliki landasan implementatif yang kuat, sekaligus memenuhi kaidah ilmiah dalam proses analisis, pengujian, dan pelaporan Tugas Akhir.

**Tabel 4.1** Kriteria Penilaian Pengumpulan Proposal Ide Analisis Data dan Video

<b>Kriteria</b>	<b>Bobot (%)</b>	<b>Deskripsi Penilaian</b>
Relevansi Ide	30%	Tingkat kesesuaian ide yang diajukan dengan business case yang diberikan serta kemampuan ide tersebut untuk diselesaikan menggunakan data yang tersedia.

<b>Kriteria</b>	<b>Bobot (%)</b>	<b>Deskripsi Penilaian</b>
Inovasi dan Kreativitas	30%	Tingkat keunikan, kreativitas, dan kebaruan solusi yang diusulkan dibandingkan dengan pendekatan konvensional yang telah ada.
Feasibility (Kelayakan Solusi)	20%	Tingkat kelayakan solusi untuk diimplementasikan dengan mempertimbangkan keterbatasan waktu, teknologi, serta ketersediaan data.
Struktur dan Kualitas Proposal	10%	Kejelasan struktur proposal, ketepatan penggunaan bahasa, serta kelogisan alur argumentasi yang disampaikan.
Presentasi Proposal	10%	Kemampuan menyampaikan ide secara jelas, sistematis, dan persuasif melalui dokumen proposal dan video pendukung.

**Tabel 4.2** Kriteria Penilaian Babak Final Jawa Timur Datathon 2025

<b>Kriteria</b>	<b>Bobot (%)</b>	<b>Deskripsi Penilaian</b>
Kualitas Eksplorasi dan Pemahaman Data	20%	Kemampuan peserta dalam memahami karakteristik dataset, melakukan pembersihan data, serta menemukan insight awal yang relevan.
Kualitas Analisis dan Interpretasi	30%	Ketepatan dalam melakukan analisis data dan menarik kesimpulan yang logis serta berbasis bukti untuk mendukung rekomendasi.

<b>Kriteria</b>	<b>Bobot (%)</b>	<b>Deskripsi Penilaian</b>
Kualitas Visualisasi Data (Dashboard)	20%	Kualitas desain dashboard yang informatif, jelas, dan interaktif, serta kemudahan visualisasi dalam menyampaikan informasi kepada pengguna.
Kualitas Rekomendasi dan Insight	20%	Tingkat relevansi dan kebermanfaatan rekomendasi yang dihasilkan berdasarkan hasil analisis data.
Presentasi dan Storytelling Data	10%	Kemampuan tim dalam menyampaikan hasil analisis dan rekomendasi secara runtut, jelas, dan menarik melalui pendekatan storytelling data.

Setelah melalui seluruh tahapan seleksi, mulai dari pengajuan proposal hingga babak final, tim JEVON ORDRICK berhasil meraih peringkat ketiga pada Jawa Timur Datathon 2025.



**Gambar 4.1** Pengumuman Juara 3 Jawa Timur Datathon 2025

Pada Gambar 4.1 ditunjukkan bahwa tim JEVON ORDRICK berhasil meraih peringkat ketiga pada Jawa Timur Datathon 2025 melalui pengembangan dashboard analisis sentimen yang bertujuan membantu pelaku UMKM dalam menganalisis persepsi masyarakat terhadap produk

atau layanan mereka berdasarkan data media sosial Instagram.

## 4.2 Hasil

### 4.2.1 Analysis

Pada tahap *analysis*, hasil yang diperoleh berupa identifikasi kebutuhan sistem *frontend* dashboard analisis sentimen UMKM yang bersifat data-driven. Analisis menunjukkan bahwa data hasil analisis sentimen yang bersumber *service ABSA (Aspect Based Sentiment Analysis)* dan Rekomendasi Konten yang di salurkan ke backend digunakan oleh beberapa komponen antarmuka secara bersamaan, seperti dashboard utama, halaman sentiment, dan halaman rekomendasi. Kondisi ini menimbulkan kebutuhan akan mekanisme pengelolaan data *frontend* yang mampu menjaga konsistensi data serta menghindari permintaan data berulang ke REST API.

Hasil analisis juga menunjukkan bahwa pengguna sistem hanya berperan sebagai konsumen informasi, sehingga kebutuhan utama sistem terletak pada stabilitas penyajian data, konsistensi antar-komponen, serta kemudahan navigasi tanpa kehilangan data. Temuan pada tahap ini menjadi dasar pemilihan arsitektur *Client Data Layer* sebagai solusi pengelolaan data *frontend*.

### 4.2.2 Requirement Specification

Hasil dari tahap requirement specification adalah tersusunnya kebutuhan fungsional dan non-fungsional sistem *frontend*. Kebutuhan fungsional mencakup kemampuan sistem untuk menampilkan ringkasan sentimen, visualisasi distribusi sentimen, data rekomendasi, serta data scraper yang bersumber dari REST API backend.

Kebutuhan non-fungsional yang dihasilkan pada tahap ini meliputi konsistensi data antar-komponen, pengendalian permintaan API, serta kemampuan sistem dalam memanfaatkan mekanisme caching. Kebutuhan tersebut menjadi acuan dalam perancangan arsitektur *frontend* dan pemilihan TanStack Query sebagai pustaka pengelolaan *Client Data Layer*.

### 4.2.3 Design

Pada tahap design, dihasilkan rancangan arsitektur *frontend* berbasis component-based architecture dengan pemisahan antara lapisan presentasi dan lapisan pengelolaan data. Hasil perancangan menunjukkan bahwa *Client Data Layer* diposisikan sebagai lapisan perantara antara REST API dan komponen antarmuka pengguna.

Selain perancangan arsitektur, tahap ini juga menghasilkan rancangan penggunaan TanStack Query sebagai implementasi *Client Data Layer*, serta Desain antarmuka untuk setiap halaman utama sistem. Rancangan tersebut menjadi acuan utama dalam proses implementasi *frontend* pada tahap selanjutnya.

## Arsitektur *Frontend*

Arsitektur *frontend* pada sistem yang diimplementasikan juga ditunjukkan melalui cara penulisan dan pengelompokan file di dalam proyek *frontend* yang mengikuti arsitektur yang diterapkan. Struktur *frontend* disusun dengan pemisahan yang jelas antara komponen *view*, logika pengelolaan data, serta definisi *type* atau struktur data dari *response* maupun *payload*, dan aturan validasi atau *schema* yang digunakan sistem. Pemisahan ini bertujuan untuk mendukung penerapan arsitektur *frontend* yang terstruktur dan mudah dipelihara.

Komponen *view* berperan sebagai lapisan antarmuka pengguna yang bertanggung jawab dalam menampilkan data dan menangani interaksi pengguna dengan sistem. Lapisan ini tidak menangani proses pengambilan data secara langsung, melainkan hanya menerima data yang telah dikelola oleh lapisan pengelolaan data. Dengan pendekatan ini, perubahan pada logika data tidak berdampak langsung terhadap tampilan antarmuka pengguna.

Lapisan logika pengelolaan data bertugas mengatur proses komunikasi dengan REST API, termasuk pengambilan data, pengelolaan status data, serta distribusi data ke komponen *view*. Lapisan ini menjadi perantara antara sumber data eksternal dan antarmuka pengguna, sehingga seluruh pengelolaan data dapat dilakukan secara terpusat dan konsisten.

Definisi *type* atau struktur data digunakan untuk merepresentasikan bentuk data yang dipertukarkan antara *frontend* dan *backend*, baik dalam bentuk *payload* permintaan maupun *response* dari server. Aturan validasi atau *schema* berfungsi untuk memastikan data yang diproses dan dikirimkan oleh sistem telah sesuai dengan format dan ketentuan yang ditetapkan. Pemisahan definisi struktur data dan aturan validasi ini membantu menjaga konsistensi data serta meminimalkan kesalahan pada proses pengolahan data.

## *Client Data Layer*

Hasil implementasi menunjukkan bahwa pengelolaan data pada sistem *frontend* berjalan secara konsisten pada seluruh halaman dashboard. Pada saat data pertama kali diakses oleh aplikasi, sistem mengambil data dari REST API dan menyimpannya sebagai bagian dari pengelolaan data internal. Data yang telah diperoleh tersebut selanjutnya digunakan kembali oleh fitur-fitur lain yang membutuhkan sumber data yang sama tanpa memicu permintaan ulang ke server. Alur pengelolaan data ini ditunjukkan pada Gambar 3.5.

Penerapan pengelolaan data terpusat ini memengaruhi perilaku sistem pada saat aplikasi digunakan. Ketika pengguna melakukan navigasi antar-halaman dan kembali ke halaman sebelumnya, data tetap ditampilkan tanpa mengalami pemulangan ulang. Perilaku ini teramatit secara konsisten pada halaman dashboard utama dan halaman dashboard sentimen yang

menggunakan sumber data yang sama. Kondisi tersebut menunjukkan bahwa data yang telah diperoleh dapat dimanfaatkan kembali selama masih relevan dengan kebutuhan sistem.

Selain itu, implementasi *Client Data Layer* juga berdampak pada efisiensi permintaan data. Ketika beberapa komponen *frontend* membutuhkan data yang sama dalam waktu yang bersamaan, sistem tidak melakukan permintaan data secara terpisah untuk setiap komponen. Sebaliknya, satu hasil pengambilan data digunakan bersama oleh seluruh komponen terkait. Perilaku ini mengurangi permintaan data yang berulang dan mendukung efisiensi komunikasi antara *frontend* dan *backend*.

Pengelolaan data melalui *Client Data Layer* juga mendukung konsistensi penyajian data antar-komponen *frontend*. Data yang digunakan oleh beberapa halaman atau komponen tetap berada pada kondisi yang selaras, sehingga tidak terjadi perbedaan informasi yang ditampilkan kepada pengguna. Dengan pendekatan ini, sistem *frontend* mampu mempertahankan konsistensi data tanpa memerlukan pengelolaan state secara manual pada setiap komponen antarmuka.

Secara keseluruhan, penerapan *Client Data Layer* menggunakan TanStack Query menghasilkan mekanisme pengelolaan data *frontend* yang lebih terstruktur, konsisten, dan efisien pada tahap implementasi sistem. Lapisan ini mendukung kebutuhan aplikasi dashboard analisis sentimen yang bersifat data-driven, di mana data yang sama digunakan oleh berbagai komponen dan perlu disajikan secara konsisten selama siklus penggunaan aplikasi.

## Desain Antarmuka

Desain antarmuka yang telah disusun pada tahap perancangan sebelumnya digunakan sebagai acuan dalam proses implementasi sistem. Perancangan tersebut bertujuan untuk memastikan penyajian website dapat ditampilkan secara terstruktur, konsisten, dan mudah dipahami oleh pengguna melalui komponen visual seperti grafik, tabel, dan indikator ringkasan data.

Pada tahap desain, rancangan antarmuka disusun menggunakan alat bantu desain seperti Figma. *Wireframe* digunakan sebagai panduan konseptual untuk menjaga konsistensi tampilan dan alur navigasi ketika sistem direalisasikan ke dalam website dengan menggunakan framework React.

Implementasi antarmuka dilakukan dengan menerjemahkan rancangan desain ke dalam komponen antarmuka yang bersifat modular dan reusable. Setiap komponen diintegrasikan dengan arsitektur *frontend* berbasis data (data-driven), di mana tampilan antarmuka berfungsi sebagai representasi dari state data yang dikelola secara terpusat. Pendekatan ini memastikan bahwa perubahan data yang diperoleh dari *backend* dapat secara otomatis tercermin pada antarmuka pengguna tanpa memerlukan pengelolaan data secara manual di setiap komponen.

Selanjutnya, arsitektur *frontend* yang telah dirancang di implementasi dengan mengintegrasikan *Client Data Layer* menggunakan TanStack Query sebagai mekanisme utama dalam pengelolaan server state. Dengan penerapan arsitektur ini, proses pengambilan, penyimpanan sementara, dan sinkronisasi data dari REST API dapat dilakukan secara terstruktur, sehingga komponen antarmuka tidak berinteraksi langsung dengan API. Implementasi ini menjadi dasar bagi pengelolaan data yang konsisten dan efisien pada seluruh bagian dashboard.

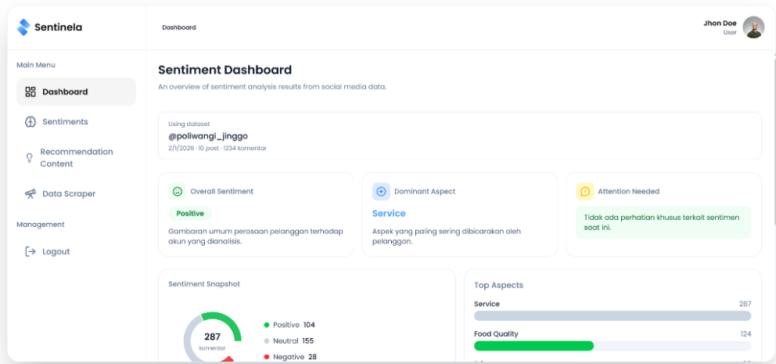
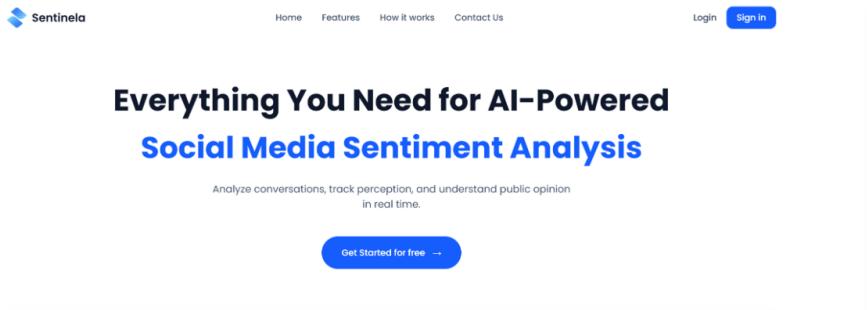
Pendekatan ini juga mendukung pemisahan tanggung jawab (separation of concerns) antara lapisan presentasi dan pengelolaan data, sehingga kompleksitas logika aplikasi dapat diminimalkan. Selain itu, arsitektur berbasis *Client Data Layer* mempermudah proses pemeliharaan dan pengembangan lanjutan sistem karena perubahan pada sumber data atau mekanisme pengambilan data tidak berdampak langsung pada komponen antarmuka.

#### **4.2.4 Coding/Implementation**

##### **Landing Page**

Landing Page diimplementasikan sebagai titik awal akses pengguna ke sistem sebelum proses autentikasi dilakukan. Pada halaman ini, *frontend* hanya menyajikan informasi umum mengenai sistem dashboard analisis sentimen tanpa melakukan pengambilan maupun pengolahan data dari backend. Oleh karena itu, halaman ini tidak melibatkan mekanisme *query* maupun *mutation* dalam pengelolaan data menggunakan TanStack Query.

Implementasi Landing Page yang bersifat statis berfungsi untuk menegaskan pemisahan antara area sistem yang tidak bersifat *data-driven* dan area sistem yang bergantung pada pengelolaan data server. Dengan pemisahan ini, aktivasi arsitektur *Client Data Layer* hanya dilakukan setelah pengguna memasuki bagian sistem yang membutuhkan interaksi dengan backend, seperti proses autentikasi dan dashboard utama. Pendekatan ini menjaga konsistensi arsitektur *frontend* agar tidak mencampurkan komponen non-data dengan mekanisme pengelolaan server state.



**Gambar 4.2** Implementasi Antarmuka Landing Page

Gambar 4.2 menunjukkan hasil implementasi antarmuka landing page. Halaman ini berfungsi sebagai halaman awal sistem yang memberikan gambaran umum mengenai aplikasi dashboard analisis sentimen. Landing page menyediakan informasi singkat terkait fungsi sistem serta navigasi menuju halaman login dan registrasi bagi pengguna. dan juga pengguna bisa mengunjungi *research version* untuk mengunjungi halaman pertama Sentinela ini ketika di perlombakan

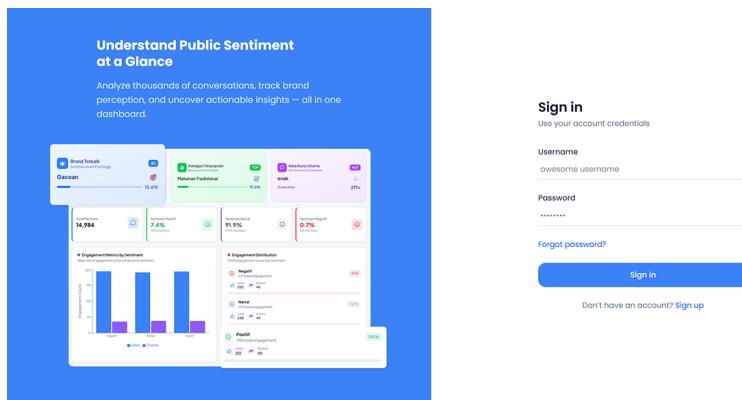
### Authentication

Fitur Authentication diimplementasikan untuk mengontrol akses pengguna terhadap sistem dashboard analisis sentimen. Proses autentikasi dilakukan melalui komunikasi *frontend* dengan REST API backend untuk memvalidasi kredensial pengguna sebelum mengizinkan akses ke fitur yang bersifat data-driven. Pada tahap ini, sistem mulai melibatkan pertukaran data antara *frontend* dan *backend* sebagai bagian dari alur penggunaan aplikasi.

Pada sisi *frontend*, hasil autentikasi menentukan status akses pengguna terhadap *Client Data Layer*. Pengambilan data menggunakan TanStack Query hanya dapat dilakukan setelah proses autentikasi berhasil, sehingga seluruh data dashboard berada dalam konteks pengguna yang valid. Implementasi ini memastikan bahwa pengelolaan data dan state aplikasi berjalan secara terkontrol serta mencegah akses data oleh pengguna yang tidak terautentikasi.

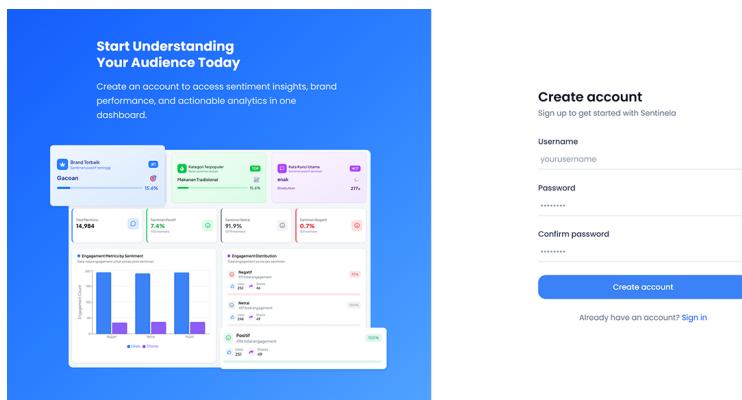
Hasil implementasi menunjukkan bahwa mekanisme autentikasi berfungsi sebagai gerbang awal sebelum *Client Data Layer* aktif digunakan. Dengan pendekatan ini, sistem

*frontend* mampu membedakan area publik dan area terproteksi secara jelas, sekaligus menjaga konsistensi pengelolaan data pada seluruh fitur yang bergantung pada hasil autentikasi.



**Gambar 4.3** Implementasi Antarmuka Halaman Login

Gambar 4.3 menunjukkan antarmuka halaman login yang digunakan untuk proses autentikasi pengguna. Pada halaman ini, pengguna dapat memasukkan kredensial berupa alamat surel dan kata sandi untuk mengakses sistem. Implementasi halaman login memastikan bahwa hanya pengguna yang terdaftar yang dapat masuk ke dalam dashboard analisis sentimen.



**Gambar 4.4** Implementasi Antarmuka Halaman Register

Gambar 4.4 menampilkan hasil implementasi halaman registrasi pengguna. Halaman ini digunakan untuk proses pendaftaran pengguna baru dengan mengisi data yang diperlukan. Data yang dimasukkan pada halaman ini selanjutnya digunakan sebagai informasi akun untuk proses autentikasi pada sistem.

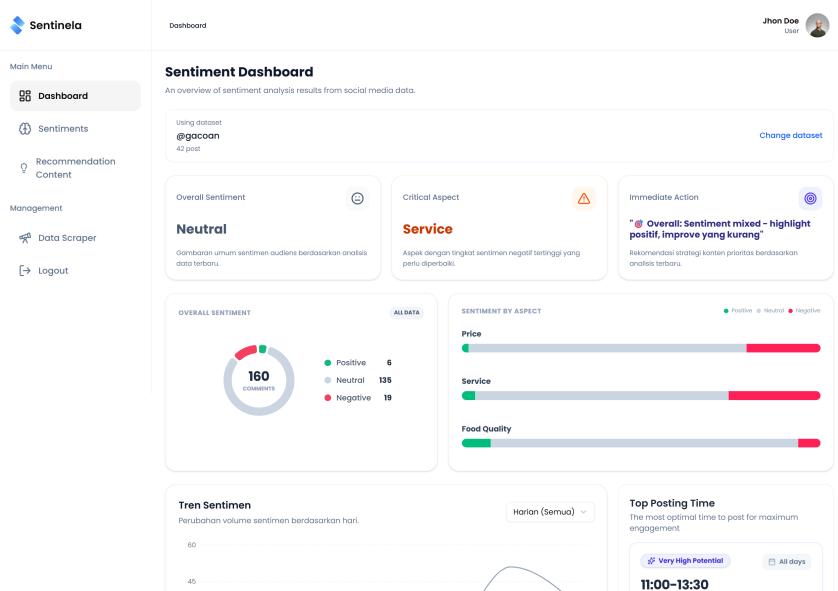
## Dashboard

Dashboard diimplementasikan sebagai halaman ringkas yang menyajikan sebagian informasi dari data sentimen dan rekomendasi konten. Data yang ditampilkan pada dashboard tidak diambil melalui *query* tersendiri, melainkan memanfaatkan data yang telah tersedia pada *Client Data Layer* dari halaman sentimen dan rekomendasi.

Pada implementasinya, dashboard berperan sebagai *consumer data* yang memetakan

sebagian data dari beberapa *query* ke dalam komponen ringkasan. Pendekatan ini memungkinkan dashboard menampilkan informasi tanpa melakukan permintaan API tambahan, karena data telah dikelola secara terpusat oleh *Client Data Layer*.

Hasil implementasi menunjukkan bahwa dashboard mampu menyajikan data yang konsisten dengan halaman sentimen dan rekomendasi. Perubahan data pada salah satu *query* secara otomatis direfleksikan pada dashboard, yang menandakan bahwa pengelolaan *server state* pada *frontend* berjalan secara terintegrasi dan efisien.



**Gambar 4.5** Implementasi Antarmuka Halaman Dashboard

Gambar 4.5 menunjukkan antarmuka halaman dashboard utama. Halaman ini berfungsi untuk menampilkan ringkasan hasil analisis sentimen dalam bentuk visualisasi dan informasi utama. Dashboard menjadi pusat akses pengguna terhadap fitur analisis sentimen dan rekomendasi konten yang tersedia pada sistem.

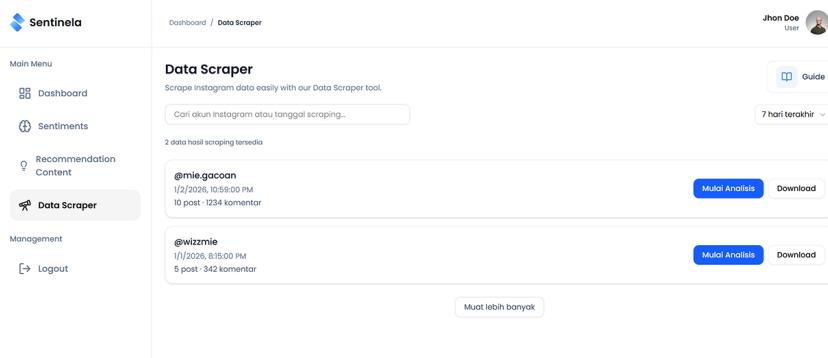
## Scrapper

Halaman Scrapper diimplementasikan untuk menampilkan daftar data hasil proses pengambilan data media sosial yang dilakukan di sisi backend. Pada halaman ini, *frontend* berperan sebagai konsumen data yang bersifat informatif dan tidak melakukan pemrosesan lanjutan terhadap data yang diterima.

Implementasi halaman Scrapper hanya melibatkan satu *query* aktif yang dikelola oleh *Client Data Layer* untuk mengambil daftar data scraping. Pendekatan ini memastikan bahwa halaman Scrapper tidak memicu pengambilan data yang tidak relevan serta tetap terpisah dari *query* milik fitur lain.

Hasil implementasi menunjukkan bahwa pengelolaan data pada halaman Scrapper berjalan

secara terkontrol, di mana hanya *query* yang sesuai dengan konteks halaman yang diaktifkan. Hal ini menegaskan bahwa *Client Data Layer* mampu membatasi ruang lingkup pengambilan data sesuai kebutuhan fitur, sehingga arsitektur *frontend* tetap tersegmentasi dengan baik.



**Gambar 4.6** Implementasi Antarmuka Halaman Scraper

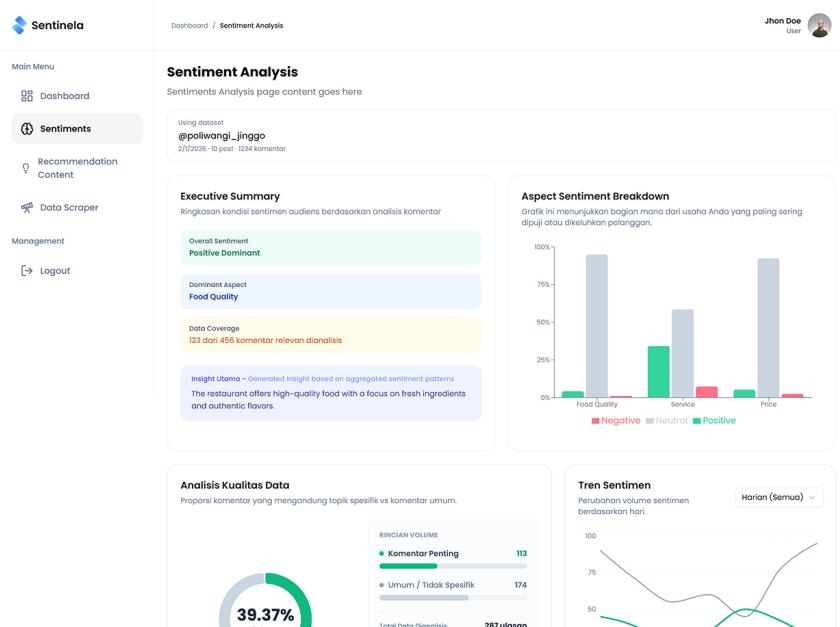
Gambar 4.6 menampilkan hasil implementasi halaman scraper. Halaman ini digunakan untuk melakukan proses pengambilan data dari sumber eksternal. Data yang berhasil dikumpulkan melalui halaman ini selanjutnya dapat digunakan sebagai bahan analisis pada sistem dashboard analisis sentimen.

### Sentiment

Halaman Sentiment diimplementasikan untuk menyajikan hasil analisis sentimen secara rinci berdasarkan data yang diperoleh dari backend. Data sentimen dikelola melalui *Client Data Layer* dan berfungsi sebagai satu sumber data yang dipercaya bagi fitur yang membutuhkan informasi sentimen, termasuk dashboard yang menampilkan ringkasan data tersebut.

Pada implementasinya, halaman Sentiment berperan sebagai consumer utama data sentimen, di mana satu query digunakan untuk mengambil dan mengelola data sentimen secara terpusat. Data yang diperoleh kemudian dipetakan ke berbagai komponen visualisasi tanpa melibatkan pengambilan data tambahan. Pendekatan ini memastikan bahwa seluruh tampilan yang memanfaatkan data sentimen mengacu pada sumber data yang sama.

Hasil implementasi menunjukkan bahwa data sentimen yang ditampilkan pada halaman Sentiment dan dashboard selalu konsisten. Setiap perubahan data pada *Client Data Layer* secara otomatis direfleksikan pada kedua halaman tersebut, yang menandakan bahwa pengelolaan server state pada *frontend* berjalan dengan baik dan mendukung penggunaan satu sumber data yang dipercaya.



**Gambar 4.7** Implementasi Antarmuka Halaman Sentiment

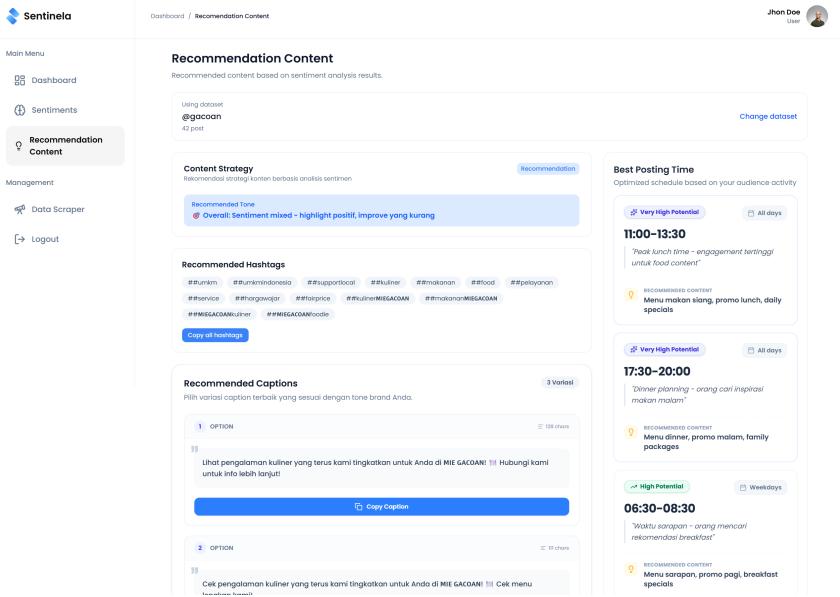
Gambar 4.7 menampilkan antarmuka halaman analisis sentimen. Halaman ini digunakan untuk menyajikan hasil analisis sentimen secara lebih rinci berdasarkan data yang dianalisis. Informasi yang ditampilkan pada halaman ini membantu pengguna dalam memahami distribusi dan kecenderungan sentimen dari data yang diproses.

### Rekomendasi Konten

Halaman rekomendasi konten diimplementasikan untuk menyajikan rekomendasi konten yang dihasilkan berdasarkan hasil analisis sentimen. Data rekomendasi diperoleh dari backend dan dikelola melalui *Client Data Layer* sebelum ditampilkan pada antarmuka pengguna. Dengan demikian, fitur ini memanfaatkan hasil analisis sebagai dasar penyusunan rekomendasi yang relevan.

Pada implementasinya, halaman rekomendasi konten berperan sebagai consumer data rekomendasi, di mana satu query digunakan untuk mengelola pengambilan dan penyajian data secara terpusat. Data yang diperoleh tidak disimpan sebagai state lokal pada komponen, melainkan diakses langsung dari *Client Data Layer*. Pendekatan ini memastikan bahwa data rekomendasi yang digunakan bersifat konsisten dan tidak terduplikasi.

Hasil implementasi menunjukkan bahwa data rekomendasi yang ditampilkan pada halaman rekomendasi konten dan dashboard selalu selaras. Setiap pembaruan data pada *Client Data Layer* secara otomatis direfleksikan pada kedua halaman tersebut, yang menandakan bahwa pengelolaan data pada *frontend* berjalan secara terintegrasi dengan mengacu pada satu sumber data yang dipercaya.



**Gambar 4.8** Implementasi Antarmuka Halaman Rekomendasi Konten

Gambar 4.8 menunjukkan antarmuka halaman rekomendasi konten. Halaman ini menyajikan hasil rekomendasi konten yang dihasilkan berdasarkan analisis sentimen data. Informasi yang ditampilkan pada halaman ini bertujuan untuk mendukung pengambilan keputusan pengguna berdasarkan hasil analisis yang tersedia.

### Chatbot

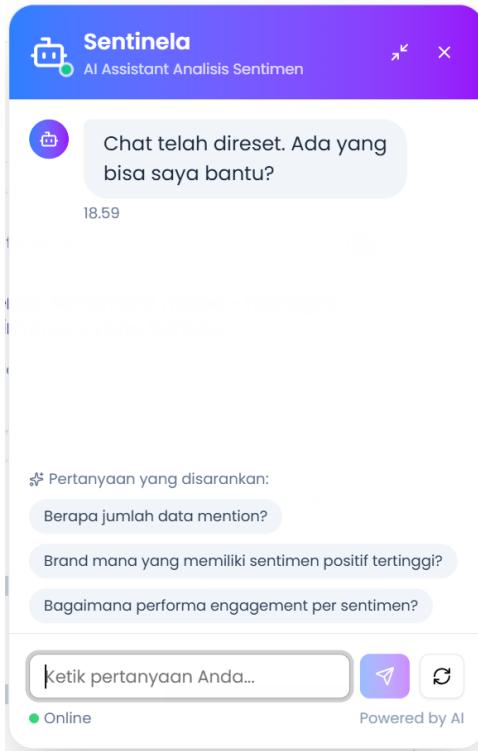
Fitur chatbot diimplementasikan sebagai komponen interaktif yang memungkinkan pengguna melakukan percakapan langsung dengan sistem untuk memperoleh informasi terkait analisis sentimen dan fungsi dashboard. Berbeda dengan halaman lain yang bersifat *read-only*, fitur chatbot melibatkan interaksi dua arah antara pengguna dan backend.

Pada sisi *frontend*, setiap pesan yang dikirimkan oleh pengguna diproses sebagai aksi yang memicu mekanisme *mutation* menggunakan TanStack Query. Data pesan pengguna dikirimkan ke backend, kemudian sistem menunggu respons chatbot secara asinkron sebelum menampilkannya kembali pada antarmuka. Pendekatan ini memungkinkan pengelolaan alur komunikasi yang terstruktur tanpa perlu melakukan pengambilan data secara berkala menggunakan *query*.

Riwayat percakapan dikelola sebagai state lokal pada komponen chatbot untuk menjaga kontinuitas interaksi selama sesi berlangsung. Sementara itu, TanStack Query berperan dalam mengelola status permintaan seperti *loading*, *success*, dan *error*, sehingga antarmuka dapat memberikan umpan balik yang jelas kepada pengguna ketika proses pengiriman pesan sedang berlangsung atau mengalami kegagalan.

Implementasi fitur chatbot ini menunjukkan penerapan *Client Data Layer* pada skenario

komunikasi interaktif berbasis permintaan pengguna. Dengan memanfaatkan mekanisme *mutation*, interaksi chatbot dapat diintegrasikan ke dalam arsitektur *frontend* secara konsisten tanpa menganggu pengelolaan server state pada fitur dashboard lainnya.



**Gambar 4.9** Desain Komponen Chatbot

Gambar 4.9 menampilkan hasil implementasi halaman chatbot. Halaman ini digunakan untuk user bisa melakukan chat dengan chatbot untuk mendapatkan informasi yang diperlukan.

#### 4.2.5 Testing

Pengujian sistem pada penelitian ini dilakukan untuk memverifikasi bahwa seluruh fitur yang telah dirancang dan diimplementasikan berfungsi sesuai dengan kebutuhan sistem yang telah ditetapkan pada Bab III. Pengujian dilakukan berdasarkan skenario pengujian yang telah disusun sebelumnya, dengan fokus pada pengujian fungsional dan perilaku sistem *frontend* dalam mengelola data serta merespons interaksi pengguna.

Pada Bab ini, hasil pengujian disajikan dalam bentuk tabel hasil pengujian yang memuat hasil aktual dari setiap skenario pengujian. Informasi mengenai data uji dan hasil yang diharapkan tidak ditampilkan kembali pada bagian ini karena telah dijelaskan secara rinci pada Bab III. Dengan demikian, penyajian hasil pengujian pada Bab ini difokuskan pada pengamatan terhadap keluaran sistem dan status keberhasilan pengujian, guna menghindari pengulangan informasi dan menekankan pada hasil empiris dari proses pengujian yang telah dilakukan. Berikut adalah hasil pengujian yang dilakukan pada sistem:

## 1. Landing Page

**Tabel 4.3** Hasil Pengujian Landing Page

ID	Skenario Pengujian	Expected Result	Hasil Aktual	Status
TC-LP-01	Pengguna membuka aplikasi tanpa melakukan proses login	Halaman landing ditampilkan dengan informasi sistem serta navigasi menuju halaman login dan registrasi.	Halaman landing berhasil ditampilkan dengan informasi sistem serta navigasi menuju halaman login dan registrasi.	Berhasil

## 2. Login

**Tabel 4.4** Hasil Pengujian Fitur Login

ID	Skenario Pengujian	Expected Result	Hasil Aktual	Status
TC-LG-01	Login dengan kredensial yang valid	Sistem memverifikasi kredensial pengguna dan mengarahkan pengguna ke halaman dashboard.	Sistem berhasil memverifikasi kredensial pengguna dan mengarahkan pengguna ke halaman dashboard.	Berhasil
TC-LG-02	Login dengan kata sandi yang salah	Sistem menampilkan pesan kesalahan autentikasi dan tetap berada pada halaman login.	Sistem menampilkan pesan kesalahan autentikasi dan pengguna tetap berada pada halaman login.	Berhasil

<b>ID</b>	<b>Skenario Pengujian</b>	<b>Expected Result</b>	<b>Hasil Aktual</b>	<b>Status</b>
TC-LG-03	Login dengan username yang tidak terdaftar	Sistem menampilkan pesan bahwa akun tidak ditemukan dan tidak melanjutkan proses login.	Sistem menampilkan pesan bahwa akun tidak ditemukan dan tidak melanjutkan proses login.	Berhasil
TC-LG-04	Login dengan field kosong	Sistem menampilkan pesan validasi bahwa seluruh field wajib diisi sebelum proses login dilakukan.	Sistem menampilkan pesan validasi bahwa seluruh field wajib diisi sebelum proses login dilakukan.	Berhasil
TC-LG-05	Login berhasil setelah percobaan gagal	Sistem menerima kredensial yang valid dan mengarahkan pengguna ke halaman dashboard.	Sistem menerima kredensial yang valid dan mengarahkan pengguna ke halaman dashboard tanpa kendala.	Berhasil

### 3. Register

**Tabel 4.5** Hasil Pengujian Fitur Register

<b>ID</b>	<b>Skenario Pengujian</b>	<b>Expected Result</b>	<b>Hasil Aktual</b>	<b>Status</b>
TC-RG-01	Registrasi dengan data yang valid	Sistem menyimpan data pengguna dan akun dapat digunakan untuk melakukan proses login.	Sistem berhasil menyimpan data pengguna dan akun dapat digunakan untuk melakukan proses login.	Berhasil

<b>ID</b>	<b>Skenario Pengujian</b>	<b>Expected Result</b>	<b>Hasil Aktual</b>	<b>Status</b>
TC-RG-02	Registrasi dengan data tidak lengkap	Sistem menampilkan pesan validasi bahwa seluruh data registrasi wajib diisi dan proses pendaftaran tidak dilanjutkan.	Sistem menampilkan pesan validasi bahwa seluruh data registrasi wajib diisi dan proses pendaftaran tidak dilanjutkan.	Berhasil
TC-RG-03	Registrasi dengan username yang sudah terdaftar	Sistem menampilkan pesan bahwa akun dengan username tersebut sudah terdaftar dan proses pendaftaran dibatalkan.	Sistem menampilkan pesan bahwa akun dengan username tersebut sudah terdaftar dan proses pendaftaran dibatalkan.	Berhasil
TC-RG-04	Registrasi ulang setelah kesalahan sebelumnya	Sistem menerima data registrasi yang valid dan proses pendaftaran berhasil diselesaikan.	Sistem menerima data registrasi yang valid dan proses pendaftaran berhasil diselesaikan.	Berhasil

#### 4. Dashboard

**Tabel 4.6** Hasil Pengujian Fitur Dashboard

<b>ID</b>	<b>Skenario Pengujian</b>	<b>Expected Result</b>	<b>Hasil Aktual</b>	<b>Status</b>
TC-DB-01	Membuka halaman dashboard setelah login	Sistem menampilkan ringkasan data dan visualisasi utama dashboard sesuai data yang tersedia.	Sistem berhasil menampilkan ringkasan data dan visualisasi utama dashboard sesuai data yang tersedia.	Berhasil

<b>ID</b>	<b>Skenario Pengujian</b>	<b>Expected Result</b>	<b>Hasil Aktual</b>	<b>Status</b>
TC-DB-02	Membuka dashboard saat data belum tersedia	Sistem menampilkan indikator pemuatan dan informasi status tanpa menimbulkan kesalahan tampilan.	Sistem menampilkan indikator pemuatan dan informasi status tanpa menimbulkan kesalahan tampilan.	Gagal
TC-DB-03	Navigasi ke menu lain dan kembali ke dashboard	Data dashboard tetap ditampilkan secara konsisten tanpa kehilangan atau perubahan data.	Data dashboard tetap ditampilkan secara konsisten tanpa kehilangan atau perubahan data.	Berhasil
TC-DB-04	Pembaruan data dashboard dari server	Sistem memperbarui tampilan dashboard sesuai dengan data terbaru yang diterima.	Sistem memperbarui tampilan dashboard sesuai dengan data terbaru yang diterima.	Berhasil
TC-DB-05	Konsistensi data antar komponen dashboard	Seluruh komponen dashboard menampilkan data yang konsisten karena mengacu pada satu sumber data.	Seluruh komponen dashboard menampilkan data yang konsisten karena mengacu pada satu sumber data.	Berhasil

## 5. Sentiment

**Tabel 4.7** Hasil Pengujian Halaman Dashboard Sentiment

ID	Skenario Pengujian	Expected Result	Hasil Aktual	Status
TC-ST-01	Membuka halaman dashboard sentiment	Sistem menampilkan visualisasi dan ringkasan hasil analisis sentimen sesuai dengan data yang tersedia.	Sistem berhasil menampilkan visualisasi dan ringkasan hasil analisis sentimen sesuai dengan data yang tersedia.	Berhasil
TC-ST-02	Membuka dashboard sentiment saat data belum tersedia	Sistem menampilkan indikator pemuatan atau pesan informasi tanpa menimbulkan kesalahan tampilan.	Sistem menampilkan indikator pemuatan atau pesan informasi tanpa menimbulkan kesalahan tampilan.	Berhasil
TC-ST-03	Pembaruan data sentimen dari server	Sistem memperbarui visualisasi dan ringkasan sentimen sesuai dengan data sentimen terbaru.	Sistem memperbarui visualisasi dan ringkasan sentimen sesuai dengan data sentimen terbaru.	Berhasil
TC-ST-04	Navigasi keluar dan kembali ke dashboard sentiment	Data sentimen tetap ditampilkan secara konsisten tanpa kehilangan atau perubahan data.	Data sentimen tetap ditampilkan secara konsisten tanpa kehilangan atau perubahan data.	Berhasil

<b>ID</b>	<b>Skenario Pengujian</b>	<b>Expected Result</b>	<b>Hasil Aktual</b>	<b>Status</b>
TC-ST-05	Konsistensi data sentimen antar komponen	Seluruh komponen visualisasi menampilkan data sentimen yang konsisten karena mengacu pada satu sumber data.	Seluruh komponen visualisasi menampilkan data sentimen yang konsisten karena mengacu pada satu sumber data.	Berhasil

## 6. Rekomendasi Konten

**Tabel 4.8** Hasil Pengujian Halaman Dashboard Rekomendasi Konten

<b>ID</b>	<b>Skenario Pengujian</b>	<b>Expected Result</b>	<b>Hasil Aktual</b>	<b>Status</b>
TC-RC-01	Membuka halaman dashboard rekomendasi konten	Sistem menampilkan daftar rekomendasi konten berdasarkan data analisis sentimen yang tersedia.	Sistem berhasil menampilkan daftar rekomendasi konten berdasarkan data analisis sentimen yang tersedia.	Berhasil
TC-RC-02	Membuka halaman rekomendasi saat data belum tersedia	Sistem menampilkan indikator pemuatan atau pesan informasi tanpa menimbulkan kesalahan tampilan.	Sistem menampilkan indikator pemuatan atau pesan informasi tanpa menimbulkan kesalahan tampilan.	Berhasil
TC-RC-03	Pembaruan data rekomendasi dari server	Sistem memperbarui daftar rekomendasi sesuai dengan data rekomendasi terbaru yang diterima.	Sistem memperbarui daftar rekomendasi sesuai dengan data rekomendasi terbaru yang diterima.	Berhasil

<b>ID</b>	<b>Skenario Pengujian</b>	<b>Expected Result</b>	<b>Hasil Aktual</b>	<b>Status</b>
TC-RC-04	Navigasi keluar dan kembali ke dashboard rekomendasi	Data rekomendasi tetap ditampilkan secara konsisten tanpa kehilangan atau perubahan data.	Data rekomendasi tetap ditampilkan secara konsisten tanpa kehilangan atau perubahan data.	Berhasil
TC-RC-05	Konsistensi rekomendasi dengan data sentimen	Rekomendasi konten menyesuaikan perubahan data sentimen yang digunakan sebagai dasar penyusunan rekomendasi.	Rekomendasi konten menyesuaikan perubahan data sentimen yang digunakan sebagai dasar penyusunan rekomendasi.	Berhasil

## 7. Scraper

**Tabel 4.9** Hasil Pengujian Halaman Data Scraper

<b>ID</b>	<b>Skenario Pengujian</b>	<b>Expected Result</b>	<b>Hasil Aktual</b>	<b>Status</b>
TC-SC-01	Membuka halaman data scraper	Sistem menampilkan daftar data hasil scraping yang diperoleh dari server.	Sistem berhasil menampilkan daftar data hasil scraping yang diperoleh dari server.	Berhasil
TC-SC-02	Membuka halaman scraper saat data belum tersedia	Sistem menampilkan indikator pemuatan atau pesan informasi tanpa menimbulkan kesalahan tampilan.	Sistem menampilkan indikator pemuatan atau pesan informasi tanpa menimbulkan kesalahan tampilan.	Berhasil

<b>ID</b>	<b>Skenario Pengujian</b>	<b>Expected Result</b>	<b>Hasil Aktual</b>	<b>Status</b>
TC-SC-03	Memilih data untuk dianalisis	Sistem menandai dataset yang dipilih dan data tersebut siap digunakan pada proses analisis.	Sistem berhasil menandai dataset yang dipilih dan data tersebut siap digunakan pada proses analisis.	Berhasil
TC-SC-04	Navigasi keluar dan kembali ke halaman data scraper	Data hasil scraping tetap ditampilkan secara konsisten tanpa kehilangan atau perubahan data.	Data hasil scraping tetap ditampilkan secara konsisten tanpa kehilangan atau perubahan data.	Berhasil

## 8. Chatbot

**Tabel 4.10** Hasil Pengujian Fitur Chatbot

<b>ID</b>	<b>Skenario Pengujian</b>	<b>Expected Result</b>	<b>Hasil Aktual</b>	<b>Status</b>
TC-CB-01	Membuka antarmuka chatbot pada dashboard	Sistem menampilkan antarmuka chatbot dan siap menerima input pengguna.	Sistem menampilkan antarmuka chatbot pada dashboard dan siap menerima input pengguna.	Berhasil
TC-CB-02	Mengirimkan pertanyaan melalui chatbot	Sistem menampilkan respons chatbot sesuai dengan pertanyaan yang diberikan.	Sistem menampilkan respons chatbot sesuai dengan pertanyaan teks yang dikirimkan.	Berhasil
TC-CB-03	Mengirimkan input kosong pada chatbot	Sistem menampilkan pesan validasi atau informasi bahwa input tidak valid.	Sistem menampilkan pesan validasi ketika pengguna mengirimkan input kosong.	Berhasil

<b>ID</b>	<b>Skenario Pengujian</b>	<b>Expected Result</b>	<b>Hasil Aktual</b>	<b>Status</b>
TC-CB-04	Mengirimkan pertanyaan di luar konteks sistem	Sistem menampilkan respons atau pesan informasi yang sesuai.	Sistem menampilkan pesan informasi yang sesuai untuk pertanyaan di luar konteks sistem.	Berhasil
TC-CB-05	Interaksi berulang dengan chatbot	Sistem mampu menampilkan respons secara konsisten pada setiap interaksi.	Sistem menampilkan respons chatbot secara konsisten pada setiap interaksi berulang.	Berhasil

#### 9. Logout

**Tabel 4.11** Hasil Pengujian Fitur Logout

<b>ID</b>	<b>Skenario Pengujian</b>	<b>Expected Result</b>	<b>Hasil Aktual</b>	<b>Status</b>
TC-LO-01	Pengguna melakukan logout dari sistem	Sistem mengakhiri sesi pengguna dan mengarahkan pengguna ke halaman login.	Sistem berhasil mengakhiri sesi pengguna dan mengarahkan pengguna ke halaman login.	Berhasil
TC-LO-02	Pengguna mencoba mengakses halaman dashboard setelah logout	Sistem menolak akses dashboard dan mengarahkan pengguna kembali ke halaman login.	Sistem menolak akses dashboard dan mengarahkan pengguna kembali ke halaman login.	Berhasil

<b>ID</b>	<b>Skenario Pengujian</b>	<b>Expected Result</b>	<b>Hasil Aktual</b>	<b>Status</b>
TC-LO-03	Pengguna menutup sesi dan membuka ulang aplikasi	Sistem meminta pengguna untuk melakukan login kembali sebelum mengakses fitur sistem.	Sistem meminta pengguna untuk melakukan login kembali sebelum mengakses fitur sistem.	Berhasil

Secara kuantitatif, hasil pengujian fungsional menunjukkan tingkat keberhasilan sebesar 97,3%, yang diperoleh dari 36 skenario pengujian berhasil dari total 37 skenario yang dirancang dan dilaksanakan. Sementara itu, tingkat kegagalan pengujian tercatat sebesar 2,7%, yang berasal dari satu skenario pengujian yang tidak sepenuhnya memenuhi *expected result*. Persentase keberhasilan ini mengindikasikan bahwa sistem telah berfungsi memenuhi sebagian besar kebutuhan fungsional yang ditetapkan, dengan kegagalan yang terjadi bersifat minor dan tidak memengaruhi fungsi utama sistem secara keseluruhan.

Satu skenario pengujian yang dinyatakan gagal terdapat pada TC-DB-02 (Membuka dashboard saat data belum tersedia). Pada skenario ini, expected result menetapkan bahwa sistem secara langsung menampilkan indikator pemuatan (loading indicator) setelah pengguna berhasil login dan mengakses halaman dashboard. Namun, hasil pengujian menunjukkan bahwa sistem terlebih dahulu menampilkan *state* data kosong dalam durasi singkat sebelum indikator pemuatan ditampilkan. Kondisi ini disebabkan oleh adanya jeda transisi *state* antara proses inisialisasi halaman dashboard dan proses pengambilan data dari server, sehingga indikator pemuatan tidak langsung muncul pada saat halaman pertama kali dirender.

Meskipun tidak menimbulkan kesalahan fungsional maupun kegagalan sistem secara keseluruhan, perilaku tersebut dinilai tidak sepenuhnya sesuai dengan *expected result*, sehingga skenario pengujian ini dikategorikan sebagai gagal.

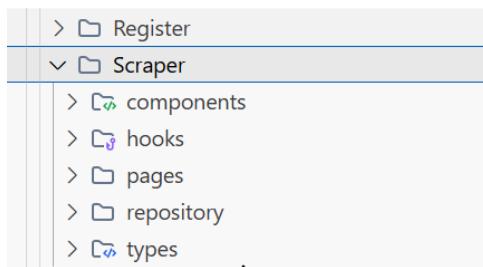
### 4.3 Pembahasan

#### 4.3.1 Peran Arsitektur *Client Data Layer* dalam Pengelolaan Data *Frontend*

Penerapan arsitektur *Client Data Layer* pada sistem yang dikembangkan berperan sebagai lapisan perantara antara REST API dan komponen antarmuka pengguna. Data hasil analisis sentimen yang diperoleh dari backend tidak diakses secara langsung oleh komponen antarmuka, melainkan dikelola terlebih dahulu melalui lapisan *Client Data Layer*. Pendekatan ini memungkinkan pemisahan tanggung jawab yang jelas antara logika pengelolaan data dan

logika presentasi antarmuka pengguna.

Berdasarkan hasil pengujian dan observasi terhadap perilaku sistem, penerapan *Client Data Layer* menunjukkan perubahan yang signifikan dalam pengelolaan data *frontend*. Data yang sama dapat digunakan secara bersama oleh beberapa komponen dashboard tanpa memicu permintaan berulang ke server. Selain itu, konsistensi data antar-komponen tetap terjaga ketika pengguna berpindah halaman atau kembali ke halaman sebelumnya, karena data dikelola dalam satu sumber kebenaran yang terpusat di sisi klien. Temuan ini menunjukkan bahwa *Client Data Layer* berperan dalam mengendalikan alur pengambilan, penyimpanan, dan distribusi data secara lebih terkontrol.



**Gambar 4.10** Struktur Folder Per Fitur

Penerapan arsitektur tersebut juga tercermin pada pengorganisasian struktur proyek. Seperti ditunjukkan pada Gambar 4.10, setiap fitur utama ditempatkan dalam direktori terpisah yang memuat lapisan-lapisan dengan tanggung jawab yang berbeda. Struktur ini digunakan untuk memastikan bahwa pengelolaan data, akses ke sumber data, dan penyajian antarmuka tidak saling bercampur secara langsung.

Dalam konteks dashboard analitik, peran *Client Data Layer* menjadi semakin relevan karena sistem bergantung pada data dinamis yang digunakan secara simultan oleh berbagai komponen visualisasi. Dengan pengelolaan data yang terpusat, sistem *frontend* mampu menyajikan informasi secara lebih stabil dan konsisten tanpa membebani backend dengan permintaan API yang tidak diperlukan. Hal ini memperkuat bahwa penerapan arsitektur *Client Data Layer* mendukung pengelolaan data *frontend* yang lebih terstruktur, khususnya dalam menjaga konsistensi data dan mengendalikan interaksi antara *frontend* dan backend.

#### 4.3.2 Analisis Implementasi TanStack Query terhadap Server State

Implementasi arsitektur *Client Data Layer* pada sistem *frontend* direalisasikan dengan memanfaatkan TanStack Query sebagai mekanisme utama dalam pengelolaan server state. Penggunaan TanStack Query memungkinkan data yang bersumber dari REST API dikelola secara terpusat di sisi klien, sehingga komponen antarmuka tidak berinteraksi langsung dengan proses pengambilan data. Dengan pendekatan ini, TanStack Query berperan sebagai fondasi

teknis yang mendukung penerapan *Client Data Layer* dalam sistem yang dikembangkan.

```
1 import { QueryClient, } from "@tanstack/react-query";
2 import { AxiosError } from "axios";
3
4 export const queryClient = new QueryClient({
5   defaultOptions: {
6     queries: {
7       staleTime: 5 * 60 * 1000,
8       gcTime: 30 * 60 * 1000,
9       refetchOnWindowFocus: false,
10      refetchOnReconnect: true,
11
12      // Retry hanya untuk error non-client
13      retry: (failureCount, error: unknown) => {
14        if (error instanceof AxiosError) {
15          const axiosError = error as AxiosError;
16          const status = axiosError?.response?.status;
17          if (status && status >= 400 && status < 500) return false;
18        }
19        return failureCount < 2;
20      },
21    },
22
23    mutations: {
24      retry: 1,
25    },
26  },
27});
```

**Gambar 4.11 Konfigurasi Query Client**

Konfigurasi global TanStack Query pada sistem ini direalisasikan melalui inisialisasi *query client* yang bisa dilihat pada Gambar 4.11, query client berfungsi sebagai pusat mekanisme pengelolaan server state pada sisi *frontend*. *query client* digunakan untuk menetapkan nilai *default options*, seperti mekanisme *caching*, *retry*, serta *refetching*, sehingga seluruh proses pengambilan dan perubahan data memiliki karakteristik yang konsisten.

```
1 //shared/query_keys.ts
2 export const scraperKeys = {
3   all: ["scraper"] as const,
4   list: () => [...scraperKeys.all, "list"] as const,
5   detail: (id: string) => [...scraperKeys.all, "detail", id] as const,
6 };
```

**Gambar 4.12 Konfigurasi Query Keys**

Dalam konteks pengambilan data, TanStack Query digunakan untuk menangani permintaan data hasil analisis sentimen dari backend melalui mekanisme *query*. Setiap data yang dikelola oleh TanStack Query diidentifikasi menggunakan *query key* yang bersifat unik. Konfigurasi dasar dapat dilihat pada Gambar 4.12. *Query key* ini berfungsi sebagai identitas data dalam mekanisme *cache*, sehingga pemisahan jenis data dan pengendalian *invalidasi* dapat

dilakukan secara lebih sistematis. Struktur *query key* yang digunakan dalam penelitian ini bisa di lihat secara lengkap pada Lampiran 1 poin 2, yang memperlihatkan bagaimana setiap *keys* dalam sistem *frontend* dikumpulkan dalam satu tempat agar mudah dalam skalabilitasnya.

Berdasarkan hasil pengujian sistem, pendekatan ini memungkinkan data yang sama digunakan secara bersama oleh beberapa komponen dashboard tanpa memicu permintaan ulang ke server. Dengan demikian, konsistensi data antar-komponen dapat terjaga, khususnya ketika pengguna berpindah halaman atau melakukan navigasi ulang pada dashboard.

```
1 import axiosClient from "@/lib/axios";
2 import type { ScraperDeleteResponse, ScraperResponse } from "../types/scraping";
3 import type { AnalyzeABSAResponse } from "../types/absa";
4
5 export const ScraperRepository = () => ({
6   get: async (): Promise<ScraperResponse> => {
7     try {
8       const response = await axiosClient.get<ScraperResponse>(
9         "/scraping/results"
10      );
11      console.log('Get response:', response.data);
12      return response.data;
13    } catch (error) {
14      console.error('Download error:', error);
15      throw error;
16    }
17  },
18},
19,
```

**Gambar 4.13 Konfigurasi Repository**

```
1 export interface ScraperResponse {
2   message: string;
3   data: Datum[];
4 }
5
6 export interface Datum {
7   id: string;
8   username: string;
9   fullname: string;
10  bio: string;
11  post_count: number;
12  is_analyzed: boolean;
13  createdAt: string; // string dari API
14 }
```

**Gambar 4.14 Konfigurasi Scraper**

Alur pengelolaan query diawali pada *repository layer*, yang bertugas menangani interaksi langsung dengan REST API. *Repository layer* berfungsi untuk memisahkan logika komunikasi data dari logika pengelolaan server state, sehingga komponen antarmuka tidak bergantung langsung pada detail pemanggilan API. Kode dari 4.15 menunjukkan bagaimana implementasi

*repository layer* yang digunakan dalam sistem ini, lebih lengkapnya dapat dilihat pada Lampiran 1 poin 3, Data yang diperoleh dari REST API selanjutnya dimodelkan menggunakan interface untuk memastikan kesesuaian struktur data sebelum dikelola oleh TanStack Query. Implementasi interface yang digunakan sebagai kontrak data ditunjukkan pada Gambar 4.16 dan dapat dilihat secara detail pada Lampiran 1 poin 4.

```
1 export const useLoginMutation = (options?: {  
2   onSuccess?: (data: LoginResponse) => void;  
3   onError?: (error: AxiosError) => void;  
4 }) => {  
5   const repo = loginRepository();  
6  
7   return useMutation<LoginResponse, AxiosError, LoginPayload>({  
8     mutationFn: repo.login,  
9     onSuccess: options?.onSuccess,  
10    onError: options?.onError,  
11  });  
12};
```

**Gambar 4.15 Konfigurasi Repository**

```
1 export const useScraperQuery = () => {  
2   const repo = ScraperRepository();  
3   const queryClient = useQueryClient();  
4   const { isAuthenticated } = useAuth();  
5  
6   const query = useQuery<ScraperResponse, Error, Scraper[]>({  
7     queryKey: scraperKeys.list(),  
8     queryFn: () => repo.get(),  
9     select: (response) => mapToScraper(response.data),  
10    staleTime: 5 * 60 * 1000, // 5 menit  
11    gcTime: 30 * 60 * 1000, // 30 menit  
12    refetchOnMount: false,  
13    refetchOnWindowFocus: false,  
14    enabled: isAuthenticated,  
15  });  
16};
```

**Gambar 4.16 Konfigurasi Scraper**

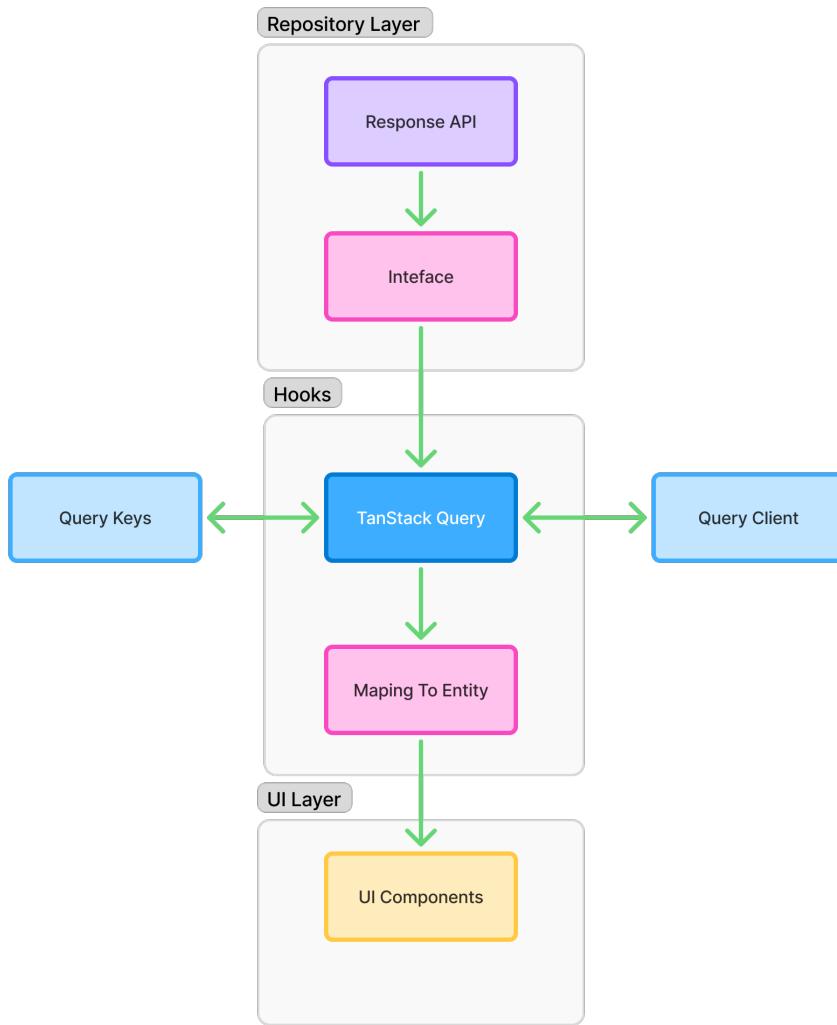
Selain mekanisme pengambilan data, pengelolaan perubahan data juga menjadi bagian penting dalam penerapan TanStack Query pada sistem *frontend* yang dikembangkan. TanStack Query menyediakan dua mekanisme utama, yaitu *query* untuk menangani pengambilan data dan *mutation* untuk menangani perubahan data yang berdampak pada kondisi sistem. Kode *Query* dan *mutation* bisa dilihat pada Gambar 4.15 dan 4.16. Kedua mekanisme ini bekerja secara terintegrasi dalam mengendalikan alur data antara *frontend* dan *backend*, baik pada fase pemuatan data awal maupun setelah terjadi perubahan data.

Implementasi mekanisme *query* pada sistem ini direalisasikan dalam bentuk *custom hook* yang mengatur proses pengambilan data dari *repository layer* serta pengelolaan *server state*

menggunakan TanStack Query. *Custom hook query* tersebut bertugas mendefinisikan *query key*, mengatur fungsi pengambilan data, serta mengelola status proses asinkron seperti *loading*, *success*, dan *error* sebelum data digunakan oleh komponen antarmuka pengguna. Implementasi *custom hook query* bisa di lihat secara lengkap pada Lampiran 1 poin 5.

Mekanisme *mutation* pada TanStack Query diimplementasikan melalui *custom hook mutation* yang bertanggung jawab dalam menangani proses perubahan data dari sisi *frontend* ke *backend*. *Custom hook mutation* ini mengelola status perubahan data serta berinteraksi dengan *query client* untuk mengendalikan pembaruan dan sinkronisasi data setelah proses *mutation* selesai. Implementasi *custom hook mutation* bisa di lihat secara lengkap pada Lampiran 1 poin 6.

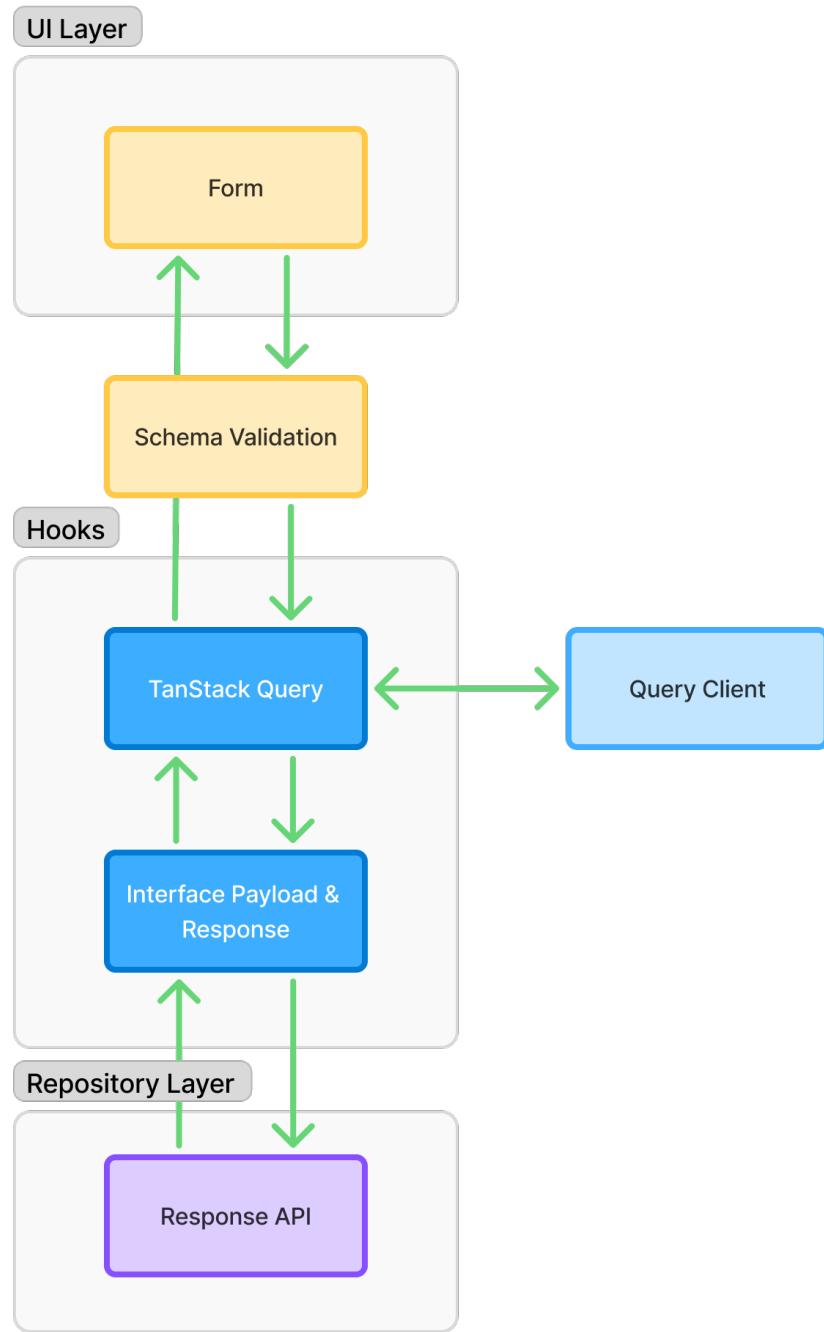
Dengan adanya pemisahan implementasi antara mekanisme *query* dan *mutation* melalui *custom hook* yang terstruktur, alur pengelolaan *server state* pada sisi *frontend* dapat diamati secara lebih sistematis. Mekanisme *query* berperan dalam menyediakan data yang dapat digunakan kembali oleh berbagai komponen antarmuka, sedangkan mekanisme *mutation* berfungsi untuk mengelola perubahan data serta sinkronisasi data setelah perubahan terjadi. Untuk memperjelas hubungan antara *repository layer*, *query client*, *custom hook*, mekanisme *query* dan *mutation*, serta komponen antarmuka pengguna dalam kedua mekanisme tersebut disajikan pada Gambar 4.17 dan Gambar 4.18.



**Gambar 4.17** Diagram Alur Pengelolaan Query menggunakan TanStack Query

Berdasarkan Gambar 4.17, terlihat alur pengelolaan *query* diawali pada *repository layer*, di mana respons data dari REST API diterima dan diproses. Data tersebut kemudian diberikan tipe data melalui *interface* sebelum dikelola pada lapisan *hooks* yang mengimplementasikan TanStack Query. Pada lapisan ini, TanStack Query mengendalikan mekanisme *caching*, *invalidasi*, dan *refetch* data. Selanjutnya, data yang telah dikelola dipetakan ke dalam bentuk *entity* untuk memastikan konsistensi struktur data sebelum dikirimkan ke lapisan presentasi atau UI.

Pendekatan ini memungkinkan komponen antarmuka pengguna mengakses data melalui satu sumber kebenaran yang terpusat, sehingga risiko inkonsistensi data antar-komponen dapat diminimalkan. Selain itu, dengan pemanfaatan mekanisme *cache* pada lapisan *query*, sistem mampu menghindari pemanggilan ulang REST API yang bersifat redundant ketika data yang sama digunakan kembali pada tampilan yang berbeda.



**Gambar 4.18** Diagram Alur Pengelolaan Mutation menggunakan TanStack Query

Berdasarkan Gambar 4.18, terlihat alur implementasi mekanisme *mutation* dalam arsitektur *Client Data Layer* menggunakan TanStack Query. Proses *mutation* diawali dari lapisan antarmuka pengguna, di mana input pengguna diproses melalui komponen formulir dan divalidasi menggunakan skema validasi untuk memastikan kesesuaian data sebelum dikirimkan ke lapisan pengelolaan data. Pendekatan ini bertujuan untuk meminimalkan kesalahan data sejak tahap awal interaksi pengguna.

Setelah proses validasi, data diteruskan ke lapisan hooks yang mengimplementasikan TanStack Query sebagai pengelola utama server state. Pada tahap ini, TanStack Query berperan

dalam mengoordinasikan pengiriman data ke backend melalui repository layer serta mengelola status eksekusi proses *mutation*. Interaksi dengan query client memungkinkan TanStack Query menerapkan konfigurasi global yang konsisten terhadap perilaku *mutation*, termasuk pengendalian pembaruan data setelah perubahan terjadi.

Respons dari backend yang diterima melalui repository layer kemudian diproses kembali pada lapisan *Client Data Layer* dengan menerapkan tipe data melalui *interface payload dan respons*. Tahapan ini memastikan bahwa data hasil *mutation* memiliki struktur yang konsisten sebelum digunakan kembali oleh sistem *frontend*. Selanjutnya, hasil pengelolaan *mutation* dikembalikan ke lapisan antarmuka pengguna untuk memperbarui tampilan dashboard sesuai dengan kondisi data terbaru.

Dengan alur tersebut, mekanisme *mutation* tidak hanya berfungsi sebagai sarana pengiriman perubahan data ke backend, tetapi juga sebagai bagian dari pengendalian konsistensi data dan sinkronisasi antara *frontend* dan backend. Implementasi ini memungkinkan antarmuka pengguna merespons setiap perubahan data secara terkontrol, sekaligus mendukung stabilitas dan kejelasan interaksi pada dashboard analitik.

Selain mekanisme pengelolaan data, TanStack Query juga menyediakan representasi state proses asinkron yang mencerminkan kondisi pemrosesan data, seperti keadaan pemuatan data (*loading*), keberhasilan proses (*success*), dan kegagalan (*error*). Representasi state ini dimanfaatkan untuk mengendalikan perilaku komponen antarmuka pengguna dalam merespons setiap kondisi tersebut. Dengan adanya informasi state yang terkelola secara terpusat, sistem *frontend* mampu menampilkan indikator pemuatan data, pesan kesalahan, serta umpan balik keberhasilan secara konsisten, sehingga meningkatkan kejelasan interaksi dan pengalaman pengguna.

Implementasi paginasi diterapkan secara spesifik pada data komentar karena data tersebut bersifat volumetrik dan berpotensi tumbuh dalam jumlah besar, berbeda dengan data ringkasan sentimen dan tren yang bersifat agregat dan berukuran relatif kecil. Pada setiap permintaan data komentar, sistem hanya menampilkan sejumlah data terbatas, yaitu 10 komentar per halaman, sehingga jumlah data yang dikelola dan dirender oleh *frontend* tetap terkendali.

Dengan pemisahan pengelolaan data tersebut, navigasi antar halaman komentar tidak memicu pemuatan ulang data ringkasan dan tren sentimen. TanStack Query memanfaatkan mekanisme *query key* dan *caching* untuk memastikan bahwa perubahan halaman hanya berdampak pada data komentar yang dipaginasi, sementara data agregat tetap diambil dari cache. Pendekatan ini memungkinkan dashboard tetap responsif meskipun jumlah total komentar yang dianalisis sangat besar, serta menunjukkan bahwa sistem memiliki batasan data yang jelas

untuk menjaga performa aplikasi.

Secara keseluruhan, implementasi TanStack Query pada sistem yang dikembangkan menunjukkan peran yang signifikan dalam merealisasikan arsitektur *Client Data Layer* secara operasional. Melalui pengelolaan mekanisme *query* dan *mutation*, TanStack Query memungkinkan pengambilan dan perubahan data dilakukan secara terpusat, konsisten, dan terkontrol. Hasil pengujian menunjukkan bahwa pendekatan ini mampu menjaga sinkronisasi data antara *frontend* dan backend, mengcegah duplikasi request ke backend, serta memastikan konsistensi data sebelum dikirimkan ke lapisan presentasi atau UI. Dengan demikian, penerapan TanStack Query tidak hanya berfungsi sebagai solusi teknis pengelolaan data, tetapi juga berkontribusi terhadap keteraturan alur data.

#### **4.3.3 Implikasi Metode Fountain terhadap Iterasi Pengembangan Dashboard**

Implikasi penerapan metode Fountain juga tercermin dari dinamika iterasi yang terjadi selama proses pengembangan dashboard. Pada tahap implementasi, beberapa aktivitas pengembangan tidak selalu dilakukan secara berurutan, melainkan berlangsung secara tumpang tindih. Sebagai contoh, pengembangan fitur autentikasi berjalan paralel dengan perancangan fitur lain, sementara proses pengujian pada halaman data scraper dilakukan bersamaan dengan pengembangan tampilan dashboard. Pola ini menunjukkan bahwa tahapan analisis, perancangan, implementasi, dan pengujian tidak terisolasi secara kaku, melainkan saling memengaruhi sepanjang siklus pengembangan.

Selain itu, perubahan kebutuhan sistem yang muncul selama pengembangan turut memicu terjadinya iterasi balik ke tahap analisis. Penyesuaian pada tampilan dashboard mengharuskan evaluasi ulang terhadap komponen yang perlu disajikan, sehingga proses analisis dan perancangan dilakukan kembali meskipun implementasi beberapa fitur telah berjalan. Kondisi ini memperlihatkan bagaimana metode Fountain mendukung fleksibilitas dalam menghadapi perubahan kebutuhan tanpa harus mengulang proses pengembangan dari awal.

Pada tahap pengujian sistem secara keseluruhan, ditemukan adanya penambahan kebutuhan fungsional berupa integrasi fitur chatbot yang perlu diimplementasikan sebelum sistem dinyatakan siap digunakan. Penambahan fitur ini mendorong terjadinya iterasi lanjutan pada tahap perancangan dan implementasi, yang kemudian diikuti dengan proses pengujian ulang. Pola iterasi semacam ini menegaskan bahwa metode Fountain memungkinkan sistem berkembang secara adaptif terhadap kebutuhan baru yang muncul selama proses pengembangan berlangsung.

#### **4.3.4 Analisis Hasil dan Temuan Pengujian Sistem**

Berdasarkan hasil pengujian integrasi sistem, ditemukan temuan penting terkait perilaku sistem pada fase awal pemuatan dashboard setelah proses autentikasi pengguna. Pada kondisi tersebut, antarmuka pengguna menampilkan *empty state* alih-alih *loading state*, meskipun proses pengambilan data scraper dari backend masih berlangsung. Perilaku ini muncul secara konsisten ketika pengguna pertama kali diarahkan ke halaman dashboard setelah login.

Hasil analisis menunjukkan bahwa temuan tersebut disebabkan oleh urutan evaluasi state pada sisi *frontend*, khususnya pada mekanisme sinkronisasi konteks dataset. Penentuan status dataset dilakukan berdasarkan kondisi data hasil pengambilan data scraper sebelum proses fetching selesai sepenuhnya. Akibatnya, kondisi data yang belum tersedia diinterpretasikan sebagai kondisi data kosong, sehingga sistem menetapkan status *empty* meskipun data masih berada dalam fase pemuatan.

Temuan ini mengindikasikan adanya keterbatasan dalam pemisahan fase siklus hidup data (data *lifecycle*) pada logika penentuan status sistem. Secara konseptual, kondisi *loading* dan *empty* merepresentasikan dua keadaan yang berbeda, namun dalam implementasi yang diuji, kedua kondisi tersebut belum dipisahkan secara eksplisit. Hal ini menyebabkan terjadinya ketidaksesuaian antara keadaan sistem yang sebenarnya dengan representasi antarmuka pengguna pada fase awal pemuatan.

Meskipun temuan tersebut tidak mengakibatkan kegagalan fungsional pada sistem, dampaknya terlihat pada pengalaman pengguna dan kejelasan interaksi awal dengan dashboard. Sistem tetap dapat menampilkan data dengan benar setelah proses pengambilan data selesai, namun representasi awal yang kurang akurat berpotensi menimbulkan kebingungan bagi pengguna. Temuan ini memberikan wawasan bahwa pengelolaan status sistem pada aplikasi dashboard yang bersifat data-driven perlu mempertimbangkan fase pemuatan data secara eksplisit agar kondisi antarmuka pengguna dapat mencerminkan keadaan sistem secara lebih tepat.

#### **4.3.5 Perbandingan dengan Penelitian Terdahulu dan Landasan Teori**

Penelitian ini tidak berfokus pada pengembangan atau evaluasi metode analisis sentimen, sebagaimana dilakukan pada penelitian *Social Media Sentiment Analysis As a New Tool for Predicting Market Trends and Consumer Behaviour* oleh (Fajarini et al., 2025). Hasil penelitian ini menunjukkan bahwa tantangan utama pada pengembangan dashboard analitik justru terletak pada pengelolaan data hasil analisis di sisi *frontend*, khususnya dalam menjaga konsistensi dan sinkronisasi data antar-komponen aplikasi.

Berbeda dengan penelitian *React Query and Lazy Loading : Performance Optimization*

*Best Practices* oleh (Micheal, 2025), hasil penelitian ini menegaskan peran TanStack Query sebagai bagian dari arsitektur Client Data Layer. Temuan pengujian menunjukkan bahwa penggunaan TanStack Query memengaruhi perilaku pengelolaan server state, terutama pada fase awal pemuatan data dan sinkronisasi data setelah perubahan terjadi.

Selain itu, pada penelitian *Effective Data Visualization Techniques for Business Decision-Makers* oleh (Rathore et al., 2025), penelitian ini menitikberatkan pada aspek desain visual dan kejelasan penyajian informasi. Penelitian ini menunjukkan bahwa efektivitas visualisasi data pada dashboard analistik sangat bergantung pada alur pengelolaan data. Tanpa pengelolaan data yang terstruktur dan konsisten, kualitas visualisasi berpotensi menurun meskipun desain visual telah dirancang dengan baik.

Dengan demikian, penelitian ini melengkapi penelitian terdahulu dengan memberikan sudut pandang pada aspek arsitektur dan pengelolaan data *frontend*. Kontribusi utama penelitian ini terletak pada analisis implikasi penerapan *Client Data Layer* dan TanStack Query terhadap perilaku sistem *frontend* dalam konteks dashboard analistik berbasis data sentimen, yang belum banyak dibahas pada penelitian sebelumnya.

#### **4.3.6 Dampak terhadap Skalabilitas dan Maintainability Sistem**

Penerapan arsitektur *Client Data Layer* dan penggunaan TanStack Query memberikan implikasi terhadap potensi skalabilitas sistem *frontend* yang dikembangkan. Dengan pengelolaan data yang terpusat pada lapisan tertentu, penambahan fitur baru yang bergantung pada data hasil analisis sentimen dapat dilakukan tanpa harus mengubah logika pengambilan data pada setiap komponen antarmuka. Pendekatan ini memungkinkan sistem *frontend* berkembang secara modular, di mana penambahan atau perluasan fitur dapat dilakukan dengan dampak minimal mungkin terhadap fitur yang sudah ada.

Dari sisi skalabilitas pengelolaan data, mekanisme caching dan pembagian server state yang diterapkan melalui TanStack Query mendukung penggunaan data secara bersama oleh beberapa komponen dashboard. Hal ini mengurangi ketergantungan terhadap permintaan data berulang ke backend ketika jumlah komponen visualisasi atau kompleksitas dashboard meningkat. Dengan demikian, sistem memiliki karakteristik yang mendukung skalabilitas fungsional, khususnya pada aplikasi dashboard analistik yang menampilkan data di berbagai komponen.

Selain skalabilitas, penerapan arsitektur yang terstruktur juga berdampak terhadap maintainability sistem. Pemisahan tanggung jawab antara lapisan akses data, pengelolaan state, dan presentasi antarmuka memudahkan proses pemeliharaan dan penelusuran kesalahan. Perubahan pada struktur API atau logika pengelolaan data dapat dilakukan pada lapisan

tertentu tanpa harus memodifikasi keseluruhan komponen antarmuka pengguna. Pendekatan ini membantu membatasi dampak perubahan dan mengurangi risiko terjadinya kesalahan yang menyebar ke berbagai bagian sistem.

Secara keseluruhan, hasil penelitian menunjukkan bahwa penerapan *Client Data Layer* dan TanStack Query memberikan kontribusi positif terhadap potensi skalabilitas dan maintainability sistem *frontend*. Meskipun penelitian ini tidak melakukan pengukuran kuantitatif terhadap performa atau beban sistem, karakteristik arsitektural yang diterapkan mendukung pengembangan sistem yang lebih terstruktur, adaptif terhadap perubahan, dan lebih mudah dipelihara dalam jangka panjang.

#### **4.3.7 Keterbatasan Penerapan TanStack Query dalam Penelitian**

TanStack Query dirancang secara khusus untuk mengelola data yang bersumber dari server, seperti proses pengambilan data, mekanisme caching, serta sinkronisasi data antara *frontend* dan *backend*. Pendekatan ini berfokus pada pengelolaan *server state*, yaitu data yang bersifat asinkron dan berasal dari sistem eksternal, sehingga tidak secara langsung mencakup pengelolaan *client state* yang bersifat lokal dan tidak bergantung pada data server.

Dalam implementasi sistem yang dikembangkan, terdapat kebutuhan pengelolaan *client state* yang berkaitan dengan konteks aplikasi, seperti pemilihan dataset aktif, pengaturan tampilan tertentu, serta pengendalian interaksi pengguna antar-halaman. Kebutuhan tersebut berada di luar ruang lingkup utama TanStack Query karena tidak melibatkan proses pengambilan atau sinkronisasi data dari server. Oleh karena itu, pengelolaan *client state* dilakukan secara terpisah dari mekanisme pengelolaan data server.

Dengan pemisahan peran tersebut, TanStack Query tetap difungsikan secara spesifik sebagai lapisan *Client Data Layer* yang bertanggung jawab atas pengelolaan data server, sementara *client state* dikelola menggunakan mekanisme internal aplikasi sesuai kebutuhan. Pendekatan ini bertujuan menjaga fokus TanStack Query pada pengelolaan data asinkron, sekaligus mencegah kompleksitas arsitektur *frontend* yang tidak diperlukan.

Temuan ini menunjukkan bahwa penerapan TanStack Query efektif untuk pengelolaan *server state* pada aplikasi dashboard analitik, namun tidak dimaksudkan untuk menggantikan seluruh kebutuhan pengelolaan state di sisi klien. Oleh karena itu, evaluasi dalam penelitian ini difokuskan pada dampak penggunaan TanStack Query terhadap konsistensi data, mekanisme caching, dan pengendalian permintaan API, tanpa melibatkan perbandingan dengan pustaka *state management* lainnya.

## BAB 5

### KESIMPULAN DAN SARAN

#### 5.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, serta pengujian sistem yang telah dilakukan pada penelitian ini, maka dapat ditarik beberapa kesimpulan sebagai berikut.

1. Penerapan arsitektur *Client Data Layer* pada pengembangan Dashboard Analisis Sentimen UMKM berhasil direalisasikan dengan memanfaatkan TanStack Query sebagai pengelola *server state*. Arsitektur ini memisahkan secara jelas antara proses pengambilan data dari backend dan komponen antarmuka pengguna, sehingga komponen *frontend* tidak berinteraksi langsung dengan REST API.
2. Penerapan TanStack Query menghasilkan mekanisme pengelolaan data *frontend* yang memanfaatkan *caching* dan penggunaan *query key* sebagai identitas data pada sisi klien. Melalui mekanisme tersebut, data yang sama dapat digunakan secara bersama oleh beberapa komponen dashboard tanpa memicu permintaan ulang ke server, sehingga konsistensi data antar-komponen dapat terjaga.

Hasil pengujian menunjukkan bahwa TanStack Query mendukung sinkronisasi data *frontend* dan backend secara terkontrol, baik pada fase pengambilan data maupun setelah terjadi perubahan data melalui mekanisme *mutation*. Pendekatan ini membantu menjaga kestabilan tampilan dashboard pada aplikasi analitik yang bersifat data-driven.

#### 5.2 Saran

Meskipun penelitian ini berhasil mencapai tujuan yang ditetapkan, masih terdapat beberapa keterbatasan yang dapat menjadi bahan pengembangan lebih lanjut. TanStack Query memiliki fokus utama pada pengelolaan *server state* dan belum dirancang untuk menangani *client state* yang bersifat lokal dan tidak bergantung pada data server. Oleh karena itu, pada implementasi sistem ini diperlukan mekanisme tambahan untuk mengelola *client state*, seperti penggunaan pustaka state management yang khusus dirancang untuk menangani *client state*.

Sebagai saran untuk penelitian selanjutnya, integrasi antara TanStack Query dan pustaka *client state* management seperti Zustand dapat dieksplorasi lebih mendalam untuk menghasilkan pengelolaan state *frontend*. Selain itu, penelitian lanjutan dapat melakukan pengujian kuantitatif terhadap performa dan skalabilitas sistem untuk mengukur dampak penerapan *Client Data Layer* dan TanStack Query secara lebih terukur pada aplikasi dashboard analitik berskala lebih besar.

*– Halaman ini sengaja dikosongkan –*

## DAFTAR PUSTAKA

- Alviani, N. A., Studi, P., Fakultas, M., and Bangsa, U. B. (2025). Transformasi Digital pada UMKM dalam Meningkatkan Daya Saing Pasar. *Master Manajemen*, 3(1):134–140.
- Aniley, D., Alemneh, E., and Abeba, G. (2024). Selection of software development life cycle models using machine learning approach. *International Journal of Computer Applications*, 186:975–8887.
- Evergreen, P. (2023). Selecting a State Management Strategy for Modern Web Frontend Applications. *Tampere University*, page 66.
- Fajarini, S., Kurniawati, J., and Yuliani, F. (2025). Social media sentiment analysis as a new tool for predicting market trends and consumer behaviour. *Proceeding of International Conference on Social Science and Humanity*, 2:899–909.
- Highlights, F. (2025). TanStack Query: A Powerful Tool for Data Management in React. Diakses pada 07 Jan 2026.
- Islam, M. M. (2025). The impact of data-driven web frameworks on performance and scalability of u.s. enterprise applications. *International Journal of Business and Economics Insights*, 05:523–558.
- Joseph, T. (2024). Natural Language Processing (NLP) for Sentiment Analysis in Social Media. *International Journal of Computing and Engineering*, 6(2):35–48.
- Luz, H. (2025). Comparing performance of redux, mobx, and react query. ResearchGate. Preprint, ResearchGate.
- Maharani, L. (2024). Analisis Pola Dan Tren Penggunaan Data Media Sosial Dengan Teknik Data Mining. *Logicloom*, 1(2):1–21.
- Mardhatilah, D., Omar, A., and Septiari, E. D. (2024). BUILDING CONSUMER ENGAGEMENT IN SOCIAL MEDIA: A SYSTEMATIC LITERATURE REVIEW. *JOURNAL OF BUSINESS MANAGEMENT AND ACCOUNTING*, 14(1):1–35.
- Maulida, M., Zahro, F., Hakim, R., and Akbar, M. S. (2025). PT. Media Akademik Publisher PENGUJIAN BLACK BOX TESTING PADA SISTEM WEBSITE PEMESANAN ONLINE TOKO AYAM KRISPY. *Jurnal Media Akademik (Jma)*, 3(5):3031–5220.
- Micheal, D. (2025). React Query and Lazy Loading: Performance Optimization Best Practices. Preprint, ResearchGate.
- Owusu, K. (2025). REST API Architecture in React A Production Guide.
- Purnama, I., Setiani, Y., Ari, F., and Wibisono, N. (2025). Analisis dan Visualisasi Data Menggunakan Looker Studio Pada Dataset New York City Property Sales. *Jurnal Minfo Polgan*, 13:2222–2234.
- Putra, F. P. E., Efendi, R. W., Tamam, A. B., and Pramadi, W. A. (2025). Trends and best practices in api-based web development using laravel and react. *Brilliance: Journal of Information and Software Engineering*, 5(1):1–10.
- Rahman, A. and Prihanto, A. (2024). Optimisasi Kinerja Aplikasi Fitness Berbasis Next.js Melalui Penerapan Metode Caching Pada PT. Anugerah Wijaya Raga. *Journal of Informatics and Computer Science (JINACS)*, 6(02):333–340.
- Rathore, S., Nawkhare, R., Sharma, N., Chaudhary, N., Chakole, S., and Vishwakrama, B. (2025). Effective data visualization techniques for business decision-makers. *International Research Journal on Advanced Engineering and Management (IRJAEM)*, 3:2029–2036.
- React (2026). React: A javascript library for building user interfaces. <https://react.dev/>. Diakses pada 07 Jan 2026.
- Sastraa, R. and Sutawinata, A. M. (2023). Perancangan Aplikasi SIP-PTK Sekolah Dasar Negeri Guntur 01 Menggunakan Model Fountain. *INSANtek*, 4(2):63–68.
- Siva, F., Assegaf, S. M. U., Pahlevi, S. A., and Yaqin, M. A. (2023). Survey metode-metode

- software development life cycle dengan metode systematic literature review. *Lembaga Penelitian dan Pengabdian Masyarakat*, 5(2).
- Sofyan, S. and Agusman (2025). ANALISIS KESIAPAN UMKM MENGHADAPI EKONOMI DIGITAL. *Smart Jurnal Ilmiah*, IX(1):1–4.
- Suryani, A. and Komputer, I. (2024). ANALISIS SENTIMEN PENGGUNA BERBASIS DATA. *Logicloom.id*, 1(4):1–21.
- Tanstack LCC (2025). Tanstack Query Official Documentation. Diakses pada 07 Jan 2026.
- Trulline, P. (2021). Pemasaran produk UMKM melalui media sosial dan e-commerce. *Jurnal Manajemen Komunikasi*, 5(2):259.

# LAMPIRAN

## Lampiran 1. *Source Code*

### 1. Query Client

```
1 import { QueryClient, } from "@tanstack/react-query";
2 import { AxiosError } from "axios";
3
4 export const queryClient = new QueryClient({
5   defaultOptions: {
6     queries: {
7       staleTime: 5 * 60 * 1000,
8       gcTime: 30 * 60 * 1000,
9       refetchOnWindowFocus: false,
10      refetchOnReconnect: true,
11
12      // Retry hanya untuk error non-client
13      retry: (failureCount, error: unknown) => {
14        if (error instanceof AxiosError) {
15          const axiosError = error as AxiosError;
16          const status = axiosError?.response?.status;
17          if (status && status >= 400 && status < 500) return false;
18        }
19        return failureCount < 2;
20      },
21    },
22
23    mutations: {
24      retry: 1,
25    },
26  },
27});
```

### 2. Query Keys

```
1 //shared/query_keys.ts
2 export const scraperKeys = {
3   all: ["scraper"] as const,
4   list: () => [...scraperKeys.all, "list"] as const,
5   detail: (id: string) => [...scraperKeys.all, "detail", id] as const,
6 };
7
8
9 export const recomendationKeys = {
10   all: ["recomendation"] as const,
11   list: () => [...recomendationKeys.all, "list"] as const,
12   detail: (id: string) => [...recomendationKeys.all, "detail", id] as const,
13 };
14
15 export const sentimentKeys = {
16   all: ["sentiment"] as const,
17   list: () => [...sentimentKeys.all, "list"] as const,
18   detail: (id: string) => [...sentimentKeys.all, "detail", id] as const,
19 };
20
21 export const dashboardKeys = {
22   all: ["dashboard"] as const,
23
24   // satu dashboard per dataset
25   detail: (datasetId: string) =>
26     [...dashboardKeys.all, "detail", datasetId] as const,
27};
```

```

28
29
30 export const insightKeys = {
31   all: ["insight"] as const,
32   list: () => [...insightKeys.all, "list"] as const,
33   detail: (id: string) => [...insightKeys.all, "detail", id] as const,
34 };
35
36
37 export const chatbotKeys = {
38   all: ["chatbot"] as const,
39   list: () => [...chatbotKeys.all, "list"] as const,
40   detail: (id: string) => [...chatbotKeys.all, "detail", id] as const,
41 };

```

### 3. Interface

```

1 export interface ScraperResponse {
2   message: string;
3   data: Datum[];
4 }
5
6 export interface Datum {
7   id: string;
8   username: string;
9   fullname: string;
10  bio: string;
11  post_count: number;
12  is_analyzed: boolean;
13  createdAt: string; // string dari API
14 }
15
16 export interface Scraper {
17   id: string;
18   username: string;
19   fullname: string;
20   bio: string;
21   post_count: number;
22   is_analyzed: boolean;
23 }
24
25 export const mapToScraper = (data: Datum[]): Scraper[] => {
26   return data.map((item) => ({
27     id: item.id,
28     username: item.username,
29     fullname: item.fullname,
30     bio: item.bio,
31     post_count: item.post_count,
32     is_analyzed: item.is_analyzed,
33   }));
34 };
35
36
37 export interface ScraperDeleteResponse {
38   message: string;
39 }

```

### 4. Repository

```

1 /* eslint-disable @typescript-eslint/no-explicit-any */
2 // Scraper.repository.tsx
3
4 import axiosClient from "@/lib/axios";
5 import type { ScraperDeleteResponse, ScraperResponse } from "../types/scrapers";

```

```

6 import type { AnalyzeABSAResponse } from "../types/absa";
7
8 export const ScraperRepository = () => ({
9
10    get: async (): Promise<ScraperResponse> => {
11        try {
12            const response = await axiosClient.get<ScraperResponse>(
13                "/scraping/results"
14            );
15            console.log('Get response:', response.data);
16            return response.data;
17        } catch (error) {
18            console.error('Download error:', error);
19            throw error;
20        }
21    },
22    // analyze absa by id
23    analyzeById: async (id: string): Promise<AnalyzeABSAResponse> => {
24        try {
25            const response = await axiosClient.post<AnalyzeABSAResponse>(
26                `/absa/${id}`,
27            );
28            return response.data;
29        } catch (error) {
30            console.error('Download error:', error);
31            throw error;
32        }
33    },
34    .....
35 });

```

## 5. Query

```

1 import { useQuery, useQueryClient } from "@tanstack/react-query";
2 import { ScraperRepository } from "../repository/scraper.repository";
3 import { mapToScraper, type Scraper } from "../types/scraper";
4 import type { ScraperResponse } from "../types/scraper";
5 import { scraperKeys } from "@/shared/query_keys";
6 import { useAuth } from "@/hooks/useAuth";
7
8 export const useScraperQuery = () => {
9    const repo = ScraperRepository();
10   const queryClient = useQueryClient();
11   const { isAuthenticated } = useAuth();
12
13   const query = useQuery<ScraperResponse, Error, Scraper[]>({
14       queryKey: scraperKeys.list(),
15       queryFn: () => repo.get(),
16       select: (response) => mapToScraper(response.data),
17
18       staleTime: 5 * 60 * 1000, // 5 menit
19       gcTime: 30 * 60 * 1000, // 30 menit
20       refetchOnMount: false,
21       refetchOnWindowFocus: false,
22       enabled: isAuthenticated,
23   });
24
25   // =====
26   // CACHE OPERATIONS
27   // =====
28
29   const setCache = (data: Scraper[]) => {
30       queryClient.setQueryData(scraperKeys.list(), data);
31   };

```

```

32 const updateOne = (updated: Scraper) => {
33   queryClient.setQueryData<Scraper[]>(
34     scraperKeys.list(),
35     (old = []) =>
36       old.map((item) =>
37         item.id === updated.id ? updated : item
38       )
39     );
40   );
41 };
42
43 const invalidate = () => {
44   queryClient.invalidateQueries({
45     queryKey: scraperKeys.list(),
46   });
47 };
48
49 const removeCache = () => {
50   queryClient.removeQueries({
51     queryKey: scraperKeys.list(),
52   });
53 };
54
55 return {
56   ...query,
57   setCache,
58   updateOne,
59   invalidate,
60   removeCache,
61 };
62 };

```

## 6. Mutation

```

1 // features/auth/hooks/useLoginMutation.ts
2 import { useMutation } from "@tanstack/react-query";
3 import { loginRepository } from "../repository/login.repository";
4 import type { LoginResponse } from "../types/login";
5 import type { LoginPayload } from "../types/login";
6 import type { AxiosError } from "axios";
7
8 export const useLoginMutation = (options?: {
9   onSuccess?: (data: LoginResponse) => void;
10  onError?: (error: AxiosError) => void;
11 }) => {
12   const repo = loginRepository();
13
14   return useMutation<LoginResponse, AxiosError, LoginPayload>({
15     mutationFn: repo.login,
16     onSuccess: options?.onSuccess,
17     onError: options?.onError,
18   });
19 };

```

## Lampiran 2. Dokumentasi Bimbingan



(a) Dokumentasi Bimbingan 1



(b) Dokumentasi Bimbingan 2



(c) Dokumentasi Bimbingan 3



(d) Dokumentasi Bimbingan 4

## Lampiran 3. Dokumentasi Penyerahan Hadiah



#### Lampiran 4. Dokumentasi dengan Dosen Pembimbing



#### Lampiran 5. Sertifikat

