

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

TUGAS AKHIR



Oleh:
NASRUL FAHMI ULUMUDDIN
NIM 362258302204

**PROGRAM STUDI SARJANA TERAPAN TEKNOLOGI REKAYASA
PERANGKAT LUNAK
JURUSAN BISNIS DAN INFORMATIKA
POLITEKNIK NEGERI BANYUWANGI
2026**

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

TUGAS AKHIR

Diajukan kepada Jurusan Bisnis dan Informatika Politeknik Negeri Banyuwangi Untuk
Memenuhi Sebagai Persyaratan Guna Memperoleh Gelar Sarjana Terapan

Oleh:

NASRUL FAHMI ULUMUDDIN

362258302204

Pembimbing:

Sepyan Purnama Kristanto, S.Kom., M.Kom

Arum Andary Ratri, S.Si., M.Si.

**PROGRAM STUDI SARJANA TERAPAN TEKNOLOGI REKAYASA
PERANGKAT LUNAK**
JURUSAN BISNIS DAN INFORMATIKA
POLITEKNIK NEGERI BANYUWANGI
2026

— Halaman ini sengaja dikosongkan —

HALAMAN PERSEMBAHAN

“Tugas Akhir ini saya persembahkan sepenuhnya kepada dua orang hebat dalam hidup saya, Ayahanda dan Ibunda. Keduanya lah yang membuat segalanya menjadi mungkin sehingga saya bisa sampai pada tahap di mana skripsi ini akhirnya selesai. Terima kasih atas segala pengorbanan, nasihat dan doa baik yang tidak pernah berhenti kalian berikan kepadaku. Aku selamanya bersyukur dengan keberadaan kalian sebagai orangtua ku.”

— Halaman ini sengaja dikosongkan —

MOTTO

"Kami tidak meragukan tamu meskipun permitaannya aneh aneh"

"Menjilat ganggang pintu adalah ilegal di planet lain"

— Halaman ini sengaja dikosongkan —

HALAMAN PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : NASRUL FAHMI ULUMUDDIN
NIM : 362258302204
Program Studi : Sarjana Terapan Teknologi Rekayasa Perangkat Lunak
Jurusan : Jurusan Bisnis dan Informatika

Menyatakan bahwa:

1. Tugas Akhir yang saya buat adalah asli dan belum pernah diajukan untuk mendapatkan gelar akademik, baik di Politeknik Negeri Banyuwangi maupun di perguruan tinggi lain.
2. Tugas Akhir ini adalah murni gagasan, rumusan, dan hasil penelitian saya sendiri dengan arahan Dosen Pembimbing.
3. Dalam Tugas Akhir ini tidak terdapat karya atau pendapat yang telah ditulis atau dipublikasikan oleh orang lain, kecuali secara tertulis dengan jelas dicantumkan sebagai acuan dalam naskah dengan disebutkan nama pengarang dan dicantumkan dalam daftar pustaka.
4. Pernyataan ini saya buat dengan sesungguhnya dan apabila di kemudian hari terdapat penyimpangan dan ketidakbenaran dalam pernyataan ini, maka saya bersedia menerima sanksi akademik berupa pencabutan gelar yang telah diperoleh karena karya ini, serta sanksi lainnya sesuai dengan norma yang berlaku di perguruan tinggi ini.

Banyuwangi, 14 Januari 2026

Yang membuat pernyataan,

NASRUL FAHMI ULUMUDDIN

NIM. 362258302204

— Halaman ini sengaja dikosongkan —

LEMBAR PERSETUJUAN

Tugas Akhir dengan Judul

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

Disusun oleh:
NASRUL FAHMI ULUMUDDIN
NIM 362258302204

telah memenuhi syarat dan disetujui oleh Dosen Pembimbing untuk dilaksanakan Ujian Tugas Akhir bagi yang bersangkutan.

Banyuwangi, 14 Januari 2026
Mengetahui,
Koordinator Program Studi, Disetujui,
Dosen Pembimbing 1,

Lutfi Hakim, S.Pd., M.T. Sepyan Purnama Kristanto, S.Kom., M.Kom
NIP. 199203302019031012 NIP. 199009052019031024

Banyuwangi, 14 Januari 2026
Disetujui,
Dosen Pembimbing 2,

Arum Andary Ratri, S.Si., M.Si.
NIP. 199209212020122021

— Halaman ini sengaja dikosongkan —

HALAMAN PENGESAHAN

Tugas Akhir

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

Disusun oleh:

NASRUL FAHMI ULUMUDDIN
NIM 362258302204

Telah dipertahankan di depan Tim Pengaji Tugas Akhir Program Studi Sarjana Terapan Teknologi Rekayasa Perangkat Lunak Jurusan Bisnis dan Informatika Politeknik Negeri Banyuwangi Pada tanggal 14 Januari 2026.

TIM PENGUJI

Nama/Jabatan	Tanda Tangan	Tanggal
Sepyan Purnama Kristanto, S.Kom., M.Kom Pembimbing 1	14 Januari 2026
Arum Andary Ratri, S.Si., M.Si. Pembimbing 2	14 Januari 2026
Devit Suwardiyanto,S.Si., M.T. Pengaji 1	14 Januari 2026
Ruth Ema Febrita, S.Pd., M.Kom. Pengaji 2	14 Januari 2026

Banyuwangi, 14 Januari 2026
Jurusan Bisnis dan Informatika, Politeknik Negeri Banyuwangi
Ketua Jurusan,

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

— Halaman ini sengaja dikosongkan —

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

Oleh
NASRUL FAHMI ULUMUDDIN
NIM: 362258302204

ABSTRAK

Abstrak adalah sebuah ringkasan singkat yang menjelaskan secara umum tentang isi dari laporan tugas akhir. Abstrak ditulis dalam tiga (3) paragraf yang berisi beberapa kalimat yang menyatakan tujuan, metode, hasil, dan kesimpulan dari laporan tugas akhir. Paragraf pertama berisi latar belakang dan tujuan tugas akhir. Paragraf kedua berisi metode dan pembahasannya. Paragraf ketiga berisi hasil dan simpulan dari tugas akhir yang dikerjakan.

Abstrak harus menjelaskan secara jelas dan singkat apa yang dibahas dalam laporan tugas akhir, mengapa penelitian ini penting dan apa yang ditemukan dari penelitian tersebut. Abstrak harus ditulis dengan bahasa yang mudah dipahami dan harus mencakup informasi penting yang dibahas dalam laporan tugas akhir.

Abstrak harus mengandung kata-kata yang relevan dengan laporan tugas akhir dan ditulis dengan bahasa yang formal dan akademik. Abstrak merupakan bagian penting dari sebuah laporan tugas akhir karena merupakan bagian yang pertama kali dibaca oleh pembaca dan harus dapat memberikan gambaran yang jelas tentang isi dari laporan tugas akhir. Oleh karena itu, abstrak harus ditulis dengan baik dan sebaik mungkin agar dapat memberikan gambaran yang jelas tentang laporan tugas akhir yang ditulis. Panjang abstrak sebaiknya dicukupkan dalam satu halaman, termasuk kata kunci. Tiga kata kunci dipandang cukup, yang masing-masingnya memuat paduan kata utama, yang dapat merepresentasikan isi Abstrak.

Kata kunci: Konsep Abstrak, Komponen Abstrak, Kata Kunci.

— Halaman ini sengaja dikosongkan —

Implementation of Client Data Layer Architecture Using TanStack Query for a Sentiment Analysis Dashboard

by:

NASRUL FAHMI ULUMUDDIN

NIM: 362258302204

ABSTRACT

The abstract is a short summary that explains in general the contents of the final assignment report. The abstract is written in three (3) paragraphs containing several sentences stating the objectives, methods, results and conclusions of the final assignment report. The first paragraph contains the background and objectives of the final assignment. The second paragraph contains the method and discussion. The third paragraph contains the results and conclusions of the final assignment carried out.

The abstract must explain clearly and concisely what is discussed in the final project report, why this research is important and what was found from the research. The abstract must be written in language that is easy to understand and must include important information discussed in the final project report.

The abstract must contain words that are relevant to the final project report and be written in formal and academic language. The abstract is an important part of a final assignment report because it is the part that is first read by the reader and must be able to provide a clear picture of the contents of the final assignment report. Therefore, the abstract must be written well and as well as possible in order to provide a clear picture of the final project report being written. The length of the abstract should be limited to one page, including keywords. Three keywords are considered sufficient, each of which contains a combination of main words, which can represent the contents of the Abstract.

Key words: Abstract Concepts, Abstract Components, Key Words.

— 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. Sepyan Purnama Kristanto, S.Kom., M.Kom dan Arum Andary Ratri, S.Si., M.Si. selaku Dosen Pembimbing TA yang telah banyak memberikan semangat, dorongan, dan bimbingan selama penyusunan Tugas Akhir ini.
2. Devit Suwardiyanto,S.Si., M.T. dan Ruth Ema Febrita, S.Pd., M.Kom. selaku Tim Penguji yang sudah memberikan koreksi perbaikan secara komprehensif terhadap TA ini.
3. Lutfi Hakim, S.Pd., M.T. selaku Koordinator Program Studi Sarjana Terapan Teknologi Rekayasa Perangkat Lunak beserta dosen dan staf yang telah memberikan bantuan dan fasilitas selama proses penyusunan pra proposal sampai dengan selesaiya TA ini.
4. Semua pihak, secara langsung maupun tidak langsung, yang tidak dapat disebutkan di sini atas bantuan dan perhatiannya selama penyusunan Tugas Akhir ini.
5. tambahkan sesuai kebutuhan
6. tambahkan sesuai kebutuhan

Akhirnya, semoga segala bantuan yang telah berikan semua pihak di atas menjadi amalan yang bermanfaat dan mendapatkan balasan dari Allah SWT dan Tugas Akhir ini menjadi informasi bermanfaat bagi pembaca atau pihak lain yang membutuhkannya.

Banyuwangi, 14 Januari 2026

NASRUL FAHMI ULUMUDDIN
362258302204

— Halaman ini sengaja dikosongkan —

DAFTAR ISI

HALAMAN SAMPUL	i
HALAMAN PERSEMBAHAN	iii
LEMBAR PERSETUJUAN UJIAN	ix
HALAMAN PENGESAHAN	xi
ABSTRAK	xiii
ABSTRACT	xv
KATA PENGANTAR	xvii
DAFTAR ISI	xix
DAFTAR SINGKATAN	xxi
DAFTAR GAMBAR	xxii
DAFTAR TABEL	xxiii
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan	2
1.4 Manfaat	3
1.5 Batasan Masalah	3
BAB 2 TINJAUAN PUSTAKA	5
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 Blackbox Testing	8
2.1.6 Arsitektur Client Data Layer	9
2.1.7 REST API	12
2.1.8 React	14
2.1.9 TanStack Query (React Query)	14
2.1.10 Metode Fountain	16
2.2 Penelitian Terkait	19
BAB 3 METODOLOGI PENELITIAN	23
3.1 Waktu dan Jadwal penelitian	23
3.1.1 Waktu Pelaksanaan Penelitian	23
3.1.2 Jadwal Kegiatan Penelitian	23
3.2 Metode Fountain	25
3.2.1 Analysis	26
3.2.2 Requirement Specification	26
3.2.3 Design	27
3.2.4 Coding (Implementation)	37
3.2.5 Testing	37
3.2.6 Operation	46
3.2.7 Maintenance	47
3.2.8 Evaluation	47

BAB 4 HASIL DAN PEMBAHASAN	49
4.1 Hasil	49
4.1. <i>Analysis</i>	49
4.1. <i>Requirement Specification</i>	49
4.1. <i>Design</i>	49
4.1. <i>Coding/Implementation</i>	57
4.1. <i>Testing</i>	57
4.1. <i>Operation</i>	57
4.1. <i>Maintenance</i>	57
4.1. <i>Evaluation</i>	58
4.2 Pembahasan	58
BAB 5 KESIMPULAN DAN SARAN	59
5.1 Kesimpulan	59
5.2 Saran	59
DAFTAR PUSTAKA	61
LAMPIRAN A KODE PROGRAM	63
Lampiran A.1. Program Pembacaan Sensor Ultrasonic	63
Lampiran A.2. Program Keseluruhan Proyek Akhir	63
LAMPIRAN B GAMBAR-GAMBAR	65
Lampiran B.1. Foto Aktivitas Kegiatan Proyek Akhir	65
Lampiran B.2. Foto Produk Proyek Akhir	65

DAFTAR SINGKATAN

DAFTAR GAMBAR

2.1	Arsitektur <i>Client Data Layer</i>	10
2.2	Fountain SDLC Model	16
3.1	Diagram Arsitektur Frontend	28
3.2	Cara Kerja TanStack Query	31
3.3	Wireframe Landing Page	32
3.4	Wireframe Halaman Login	33
3.5	Wireframe Halaman Register	33
3.6	Wireframe Halaman Dashboard	34
3.7	Wireframe Halaman Sentiment	35
3.8	Wireframe Halaman Scraper	36
3.9	Wireframe Halaman Recomendation	36
4.1	Desain Halaman Landing Page	52
4.2	Desain Halaman Login	53
4.3	Desain Halaman Register	53
4.4	Desain Halaman Dashboard	54
4.5	Desain Halaman Sentiment	55
4.6	Desain Halaman Recommendation Content	56
4.7	Desain Halaman Scraper	56

DAFTAR TABEL

2.1	Tabel Perbandingan Penelitian Terkait	19
3.1	Jadwal Kegiatan Penelitian	23
3.2	Skenario Pengujian Landing Page	38
3.3	Skenario Pengujian Halaman Login	38
3.4	Skenario Pengujian Halaman Register	39
3.5	Skenario Pengujian Halaman Dashboard	41
3.6	Skenario Pengujian Halaman Dashboard Sentiment	42
3.7	Skenario Pengujian Halaman Dashboard Rekomendasi Konten	43
3.8	Skenario Pengujian Halaman Data Scraper	44
3.9	Skenario Pengujian Chatbot	45
3.10	Skenario Pengujian Logout	46

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 dan diperbarui secara berkala melalui API. Jika pengambilan data dilakukan secara langsung pada setiap komponen tanpa arsitektur pengelolaan data yang terstruktur, maka dapat muncul berbagai permasalahan, seperti permintaan API berulang, inkonsistensi data antar-komponen, serta kesulitan dalam pemeliharaan kode aplikasi.

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* berbasis log.

1.2 Rumusan Masalah

1. Bagaimana menerapkan arsitektur *Client Data Layer* menggunakan TanStack Query pada pengembangan Dashboard Analisis Sentimen UMKM?
2. Bagaimana perilaku pengelolaan data frontend yang dihasilkan setelah penerapan TanStack Query ditinjau dari log permintaan data, 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 perilaku 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* dengan mengamati log sistem, tanpa melakukan pengujian performa numerik atau *benchmarking* mendalam.
6. Hasil evaluasi disajikan dalam bentuk ringkasan log, tabel, dan visualisasi sederhana

untuk menggambarkan perilaku sistem setelah penerapan TanStack Query.

7. 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 & 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 bernalih 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. Pendekatan ini relevan karena media sosial menyediakan data opini konsumen yang bersifat spontan, terbuka, dan dihasilkan secara terus-menerus melalui interaksi pengguna.

Seiring dengan meningkatnya aktivitas pengguna di media sosial, volume data opini yang dihasilkan juga semakin besar dan bersifat dinamis. Data tersebut umumnya tidak terstruktur dan 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 ringkas dan terstruktur, sehingga dapat digunakan untuk memahami kecenderungan opini publik dan perilaku konsumen secara lebih sistematis.

Penelitian yang dilakukan oleh (Fajarini et al., 2025) menunjukkan bahwa analisis sentimen berbasis data media sosial merupakan metode yang inovatif dan efektif dalam memprediksi tren pasar serta memahami perilaku konsumen. Dengan memanfaatkan data berskala besar yang dihasilkan secara real time oleh pengguna media sosial, analisis sentimen mampu mengenali pola opini publik, preferensi konsumen, serta perubahan tren yang relevan di berbagai sektor industri. Hasil penelitian tersebut juga menegaskan bahwa informasi yang dihasilkan dari analisis sentimen dapat memberikan wawasan yang bernilai untuk mendukung pengambilan keputusan bisnis berbasis data.

Lebih lanjut, (Fajarini et al., 2025) menyatakan bahwa analisis sentimen berbasis media sosial menawarkan pendekatan yang lebih cepat dan efisien dibandingkan metode konvensional, seperti survei manual atau riset pasar tradisional, karena memungkinkan pemantauan opini publik secara berkelanjutan. Dengan demikian, analisis sentimen dipandang sebagai alat yang penting dalam mendukung strategi bisnis modern yang berbasis data dan responsif terhadap dinamika pasar.

Dalam penelitian ini, analisis sentimen diposisikan sebagai proses pengolahan data yang dilakukan 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*. Hasil analisis sentimen tersebut digunakan sebagai input utama untuk mendukung penyajian informasi yang informatif, terstruktur, dan mudah dipahami oleh pengguna.

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.

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.

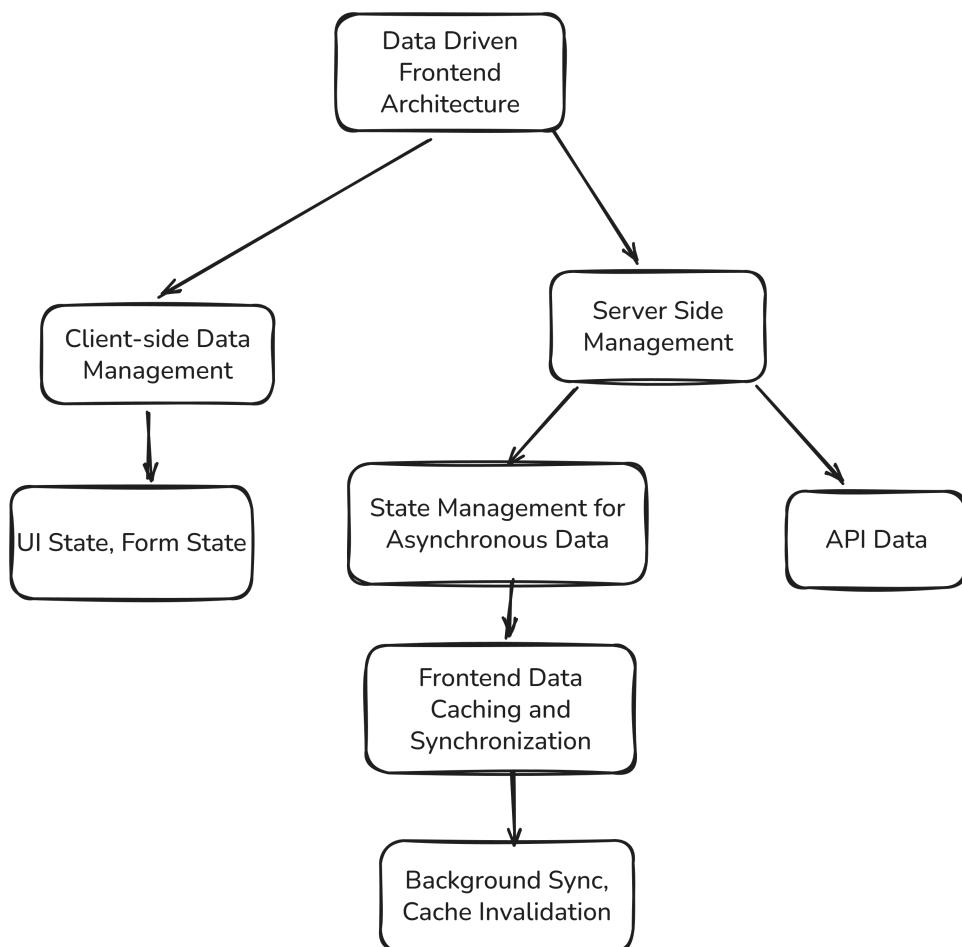
Berdasarkan karakteristik tersebut, black-box testing dipandang sebagai metode pengujian yang relevan untuk digunakan dalam penelitian ini. Metode ini memungkinkan evaluasi terhadap perilaku sistem frontend secara menyeluruh berdasarkan skenario penggunaan, sehingga kesesuaian fungsi sistem dengan kebutuhan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

2. *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, sehingga dapat mengurangi ketergantungan terhadap permintaan data berulang ke server dan menjaga konsistensi informasi yang ditampilkan. 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.

3. *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, atau aplikasi dengan data real-time. 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.

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 & Prihanto, 2024). Dalam konteks penelitian ini, peningkatan performa dipahami sebagai perubahan perilaku pengelolaan data *frontend* yang diamati melalui mekanisme *caching* dan pengendalian permintaan data, tanpa dilakukan pengukuran performa numerik.

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.

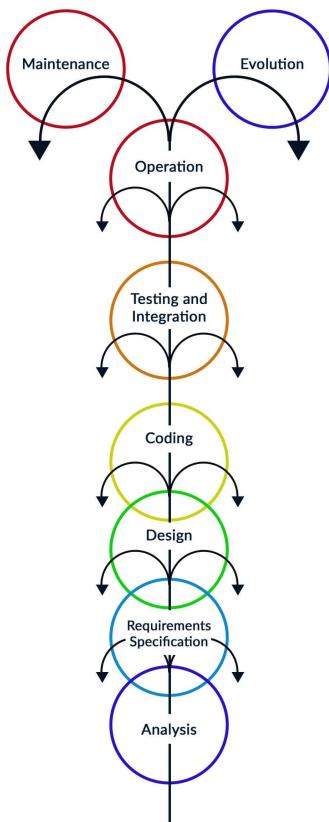
Dalam konteks arsitektur frontend, TanStack Query dapat diposisikan sebagai implementasi konkret dari *Client Data Layer*. Pustaka ini berperan sebagai lapisan perantara antara backend API dan komponen antarmuka pengguna, sehingga komponen UI tidak berinteraksi langsung dengan API. Dengan demikian, TanStack 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.



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 evaluation 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 *evaluation* merupakan tahapan evaluasi terhadap keseluruhan proses dan hasil pengembangan sistem. Evaluasi dilakukan untuk menilai apakah sistem telah memenuhi kebutuhan yang ditetapkan serta untuk mengidentifikasi kemungkinan perbaikan atau pengembangan lebih lanjut di masa mendatang.

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 & 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.

No	Peneliti	Teknologi	Judul	Fitur
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 frontend 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 dan evaluasi. 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 Kegiatan Penelitian

No	Nama Kegiatan	Bulan					
		Ags	Sep	Okt	Nov	Des	Jan
1	Analisis Kebutuhan Sistem						
2	Perancangan Arsitektur Frontend dan Client Data Layer						
3	Desain Antarmuka dan Desain Sistem						
4	Implementasi Frontend dan TanStack Query						

No	Nama Kegiatan	Bulan					
		Ags	Sep	Okt	Nov	Des	Jan
5	Testing						
6	Analisis Hasil Pengujian dan Penyusunan Laporan						

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 dievaluasi dan 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.

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 evaluasi. 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 & 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. 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 secara berkala agar informasi yang ditampilkan tetap relevan dengan kondisi terbaru tanpa memerlukan interaksi manual yang berlebihan dari pengguna. 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.

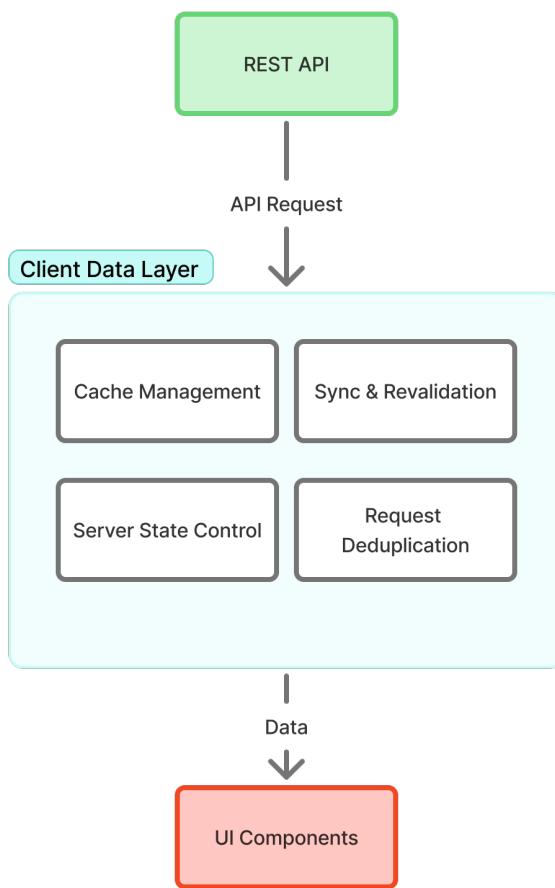
3.2.3 *Design*

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 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. 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 3.1 Diagram Arsitektur Frontend

Diagram arsitektur frontend pada Gambar 3.1 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.

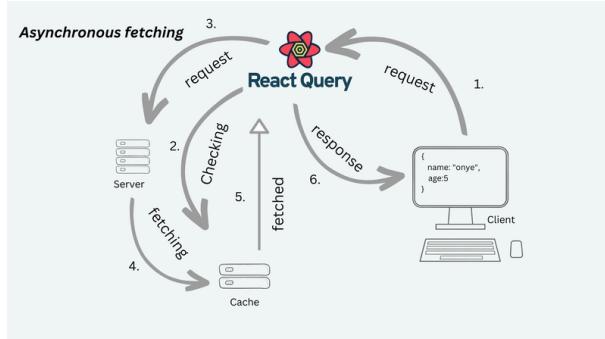
2. Perancangan Client Data Layer

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 3.1 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.

3. 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.

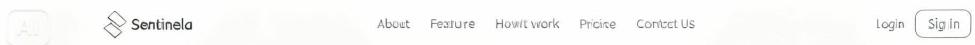


Gambar 3.2 Cara Kerja TanStack Query

Gambar 3.2 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.

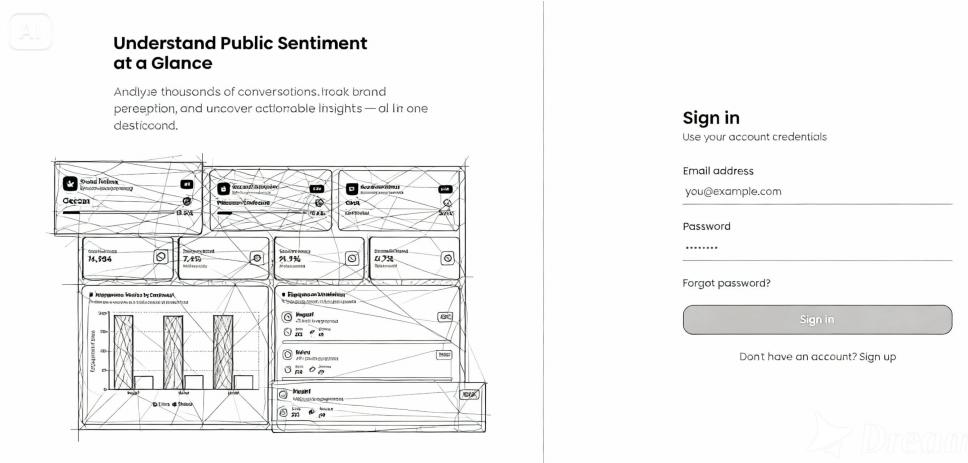
4. Perancangan Antarmuka

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. Wireframe digunakan sebagai rancangan awal antarmuka sebelum tahap implementasi, sehingga pengembangan sistem dapat dilakukan secara terstruktur dan selaras dengan kebutuhan pengguna yang telah dianalisis pada tahap sebelumnya.



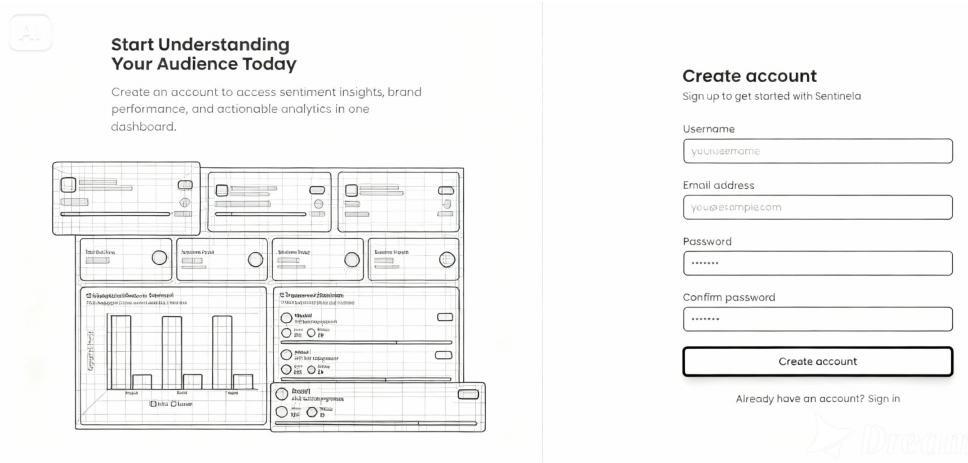
Gambar 3.3 Wireframe Landing Page

Landing Page pada Gambar 3.3 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 3.4 Wireframe Halaman Login

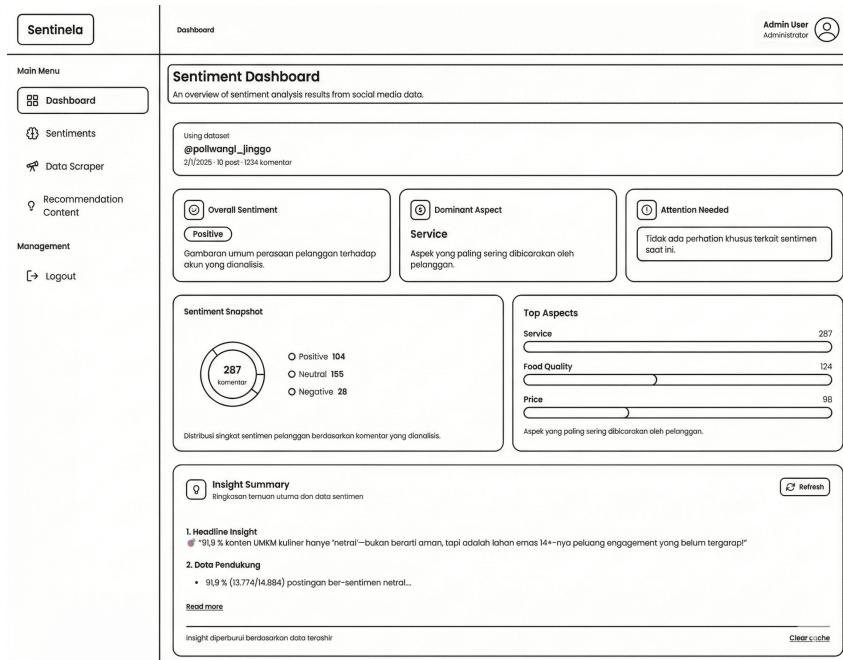
Halaman Login pada Gambar 3.4 dirancang sebagai antarmuka autentikasi pengguna untuk mengakses fitur utama sistem. Halaman ini menyediakan elemen input untuk memasukkan kredensial pengguna untuk memproses autentikasi. Pengguna dapat melakukan proses login sebelum diarahkan ke halaman dashboard. Halaman ini dirancang untuk mendukung keamanan akses sistem dengan memastikan bahwa hanya pengguna yang telah terautentikasi yang dapat mengakses fitur-fitur utama aplikasi.



Gambar 3.5 Wireframe Halaman Register

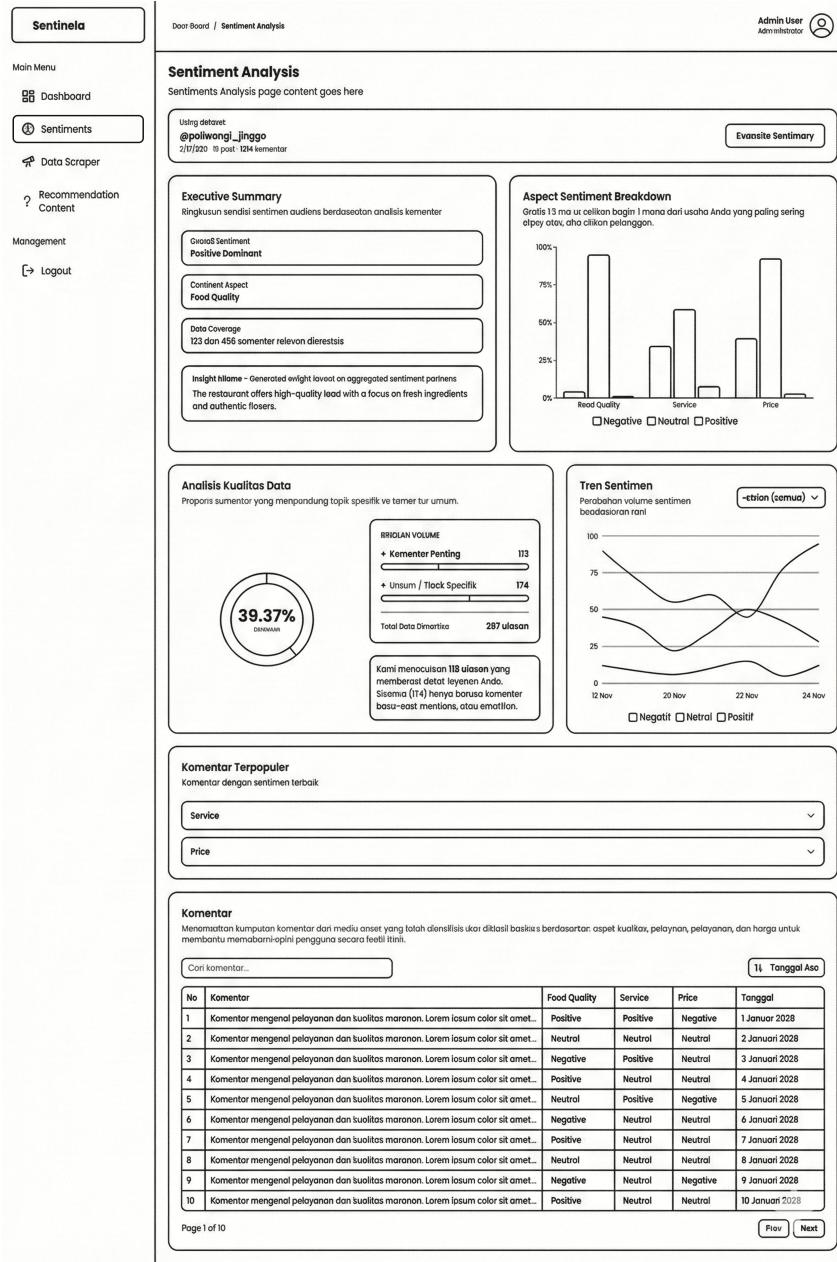
Halaman Register pada Gambar 3.5 dirancang sebagai antarmuka pendaftaran pengguna baru sebelum dapat mengakses sistem. Halaman ini menyediakan form untuk pengisian data pengguna yang diperlukan dalam proses registrasi. Pengguna dapat melakukan proses pendaftaran secara sistematis. Halaman ini juga berfungsi sebagai bagian dari mekanisme kontrol akses dengan memastikan bahwa data pengguna dikumpulkan dan diproses sebelum akun dapat digunakan untuk

mengakses fitur utama sistem.



Gambar 3.6 Wireframe Halaman Dashboard

Halaman Dashboard pada Gambar 3.6 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.



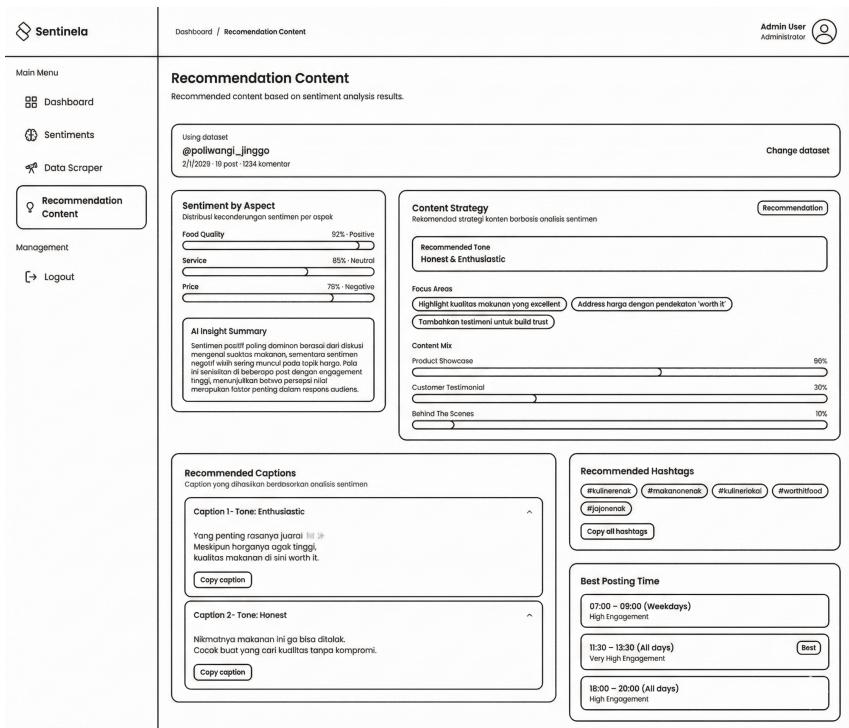
Gambar 3.7 Wireframe Halaman Sentiment

Halaman Sentiment pada Gambar 3.7 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 3.8 Wireframe Halaman Scraper

Halaman Scraper pada Gambar 3.8 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 3.9 Wireframe Halaman Recomendation

Halaman Recommendation pada Gambar 3.9 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.

3.2.4 Coding (Implementation)

Pada tahap persiapan implementasi, dilakukan penyiapan lingkungan pengembangan frontend serta penyesuaian struktur proyek agar sesuai dengan rancangan arsitektur yang telah ditetapkan. Framework React dipilih sebagai dasar pengembangan antarmuka pengguna, sementara TanStack Query digunakan sebagai mekanisme pengelolaan server state dan Client Data Layer pada aplikasi frontend. Selain itu, pada tahap ini juga dilakukan penyiapan komponen pendukung dan mekanisme integrasi data dengan backend melalui API. Persiapan tersebut bertujuan untuk memastikan proses implementasi dapat berjalan secara terstruktur, konsisten, dan sesuai dengan 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 yang telah dijelaskan pada Bab II, 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 evaluasi 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 evaluasi 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	Fitur	Skenario Pengujian	Data Uji	Hasil yang Diharapkan
TC-LP-01	Landing Page	Pengguna membuka aplikasi tanpa melakukan proses login	–	Halaman landing tampil dengan informasi sistem serta navigasi menuju halaman login dan registrasi.

2. Login

Tabel 3.3 Skenario Pengujian Halaman Login

ID	Fitur	Skenario Pengujian	Data Uji	Hasil yang Diharapkan
TC-LG-01	Login	Pengguna melakukan login dengan kredensial yang valid	Email dan kata sandi valid	Sistem berhasil mengautentikasi pengguna dan mengarahkan ke halaman dashboard
TC-LG-02	Login	Pengguna melakukan login dengan kata sandi yang salah	Email valid, kata sandi salah	Sistem menampilkan pesan kesalahan dan tetap berada di halaman login

ID	Fitur	Skenario Pengujian	Data Uji	Hasil yang Diharapkan
TC-LG-03	Login	Pengguna melakukan login dengan email yang tidak terdaftar	Email tidak terdaftar	Sistem menampilkan pesan bahwa akun tidak ditemukan
TC-LG-04	Login	Pengguna mengirimkan formulir login dengan field kosong	Email atau kata sandi kosong	Sistem menampilkan validasi bahwa seluruh field wajib diisi
TC-LG-05	Login	Pengguna berhasil login setelah sebelumnya gagal login	Kredensial valid	Sistem menerima kredensial dan mengarahkan pengguna ke halaman dashboard

3. Register

Tabel 3.4 Skenario Pengujian Halaman Register

ID	Fitur	Skenario Pengujian	Data Uji	Hasil yang Diharapkan
TC-RG-01	Register	Pengguna melakukan pendaftaran dengan data yang valid	Data registrasi lengkap dan valid	Sistem berhasil menyimpan data pengguna dan akun dapat digunakan untuk login

ID	Fitur	Skenario Pengujian	Data Uji	Hasil yang Diharapkan
TC-RG-02	Register	Pengguna melakukan pendaftaran dengan data tidak lengkap	Salah satu atau beberapa field kosong	Sistem menampilkan pesan validasi bahwa data registrasi harus diisi lengkap
TC-RG-03	Register	Pengguna melakukan pendaftaran dengan format data tidak sesuai	Format email tidak valid	Sistem menampilkan pesan kesalahan format data
TC-RG-04	Register	Pengguna mendaftarkan akun dengan email yang sudah terdaftar	Email sudah terdaftar	Sistem menampilkan pesan bahwa akun sudah terdaftar
TC-RG-05	Register	Pengguna mengirimkan ulang data registrasi setelah terjadi kesalahan	Data registrasi valid	Sistem menerima data dan proses registrasi berhasil

4. Dashboard

Tabel 3.5 Skenario Pengujian Halaman Dashboard

ID	Fitur	Skenario Pengujian	Data Uji	Hasil yang Diharapkan
TC-DB-01	Dashboard	Pengguna membuka halaman dashboard setelah login	Data dashboard tersedia	Sistem menampilkan ringkasan data dan visualisasi utama pada dashboard
TC-DB-02	Dashboard	Pengguna membuka dashboard dengan data belum tersedia	Data belum tersedia	Sistem menampilkan indikator pemuatan atau pesan informasi
TC-DB-03	Dashboard	Pengguna melakukan navigasi ke menu lain dan kembali ke dashboard	–	Sistem menampilkan data dashboard secara konsisten tanpa kehilangan data
TC-DB-04	Dashboard	Pembaruan data dashboard dari server	Data diperbarui	Sistem memperbarui tampilan dashboard sesuai data terbaru
TC-DB-05	Dashboard	Beberapa komponen menggunakan sumber data yang sama	Data yang sama digunakan	Sistem menampilkan data yang konsisten pada seluruh komponen dashboard

5. Sentiment

Tabel 3.6 Skenario Pengujian Halaman Dashboard Sentiment

ID	Fitur	Skenario Pengujian	Data Uji	Hasil yang Diharapkan
TC-ST-01	Dashboard Sentiment	Pengguna membuka halaman dashboard sentiment	Data sentimen tersedia	Sistem menampilkan visualisasi dan ringkasan hasil analisis sentimen
TC-ST-02	Dashboard Sentiment	Pengguna membuka halaman sentiment dengan data belum tersedia	Data sentimen belum tersedia	Sistem menampilkan indikator pemuatan atau pesan informasi
TC-ST-03	Dashboard Sentiment	Pembaruan data sentimen dari server	Data sentimen diperbarui	Sistem memperbarui visualisasi sesuai data sentimen terbaru
TC-ST-04	Dashboard Sentiment	Navigasi ke halaman lain dan kembali ke dashboard sentiment	–	Sistem menampilkan data sentimen secara konsisten tanpa kehilangan data
TC-ST-05	Dashboard Sentiment	Beberapa komponen menampilkan data sentimen yang sama	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
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	Fitur	Skenario Pengujian	Data Uji	Hasil yang Diharapkan
TC-SC-01	Data Scraper	Pengguna membuka halaman data scraper	Data hasil scraping tersedia	Sistem menampilkan daftar data hasil scraping yang diperoleh dari server
TC-SC-02	Data Scraper	Pengguna membuka halaman scraper dengan data belum tersedia	Data belum tersedia	Sistem menampilkan indikator pemuatan atau pesan informasi
TC-SC-03	Data Scraper	Pengguna memilih data untuk dianalisis	Dataset tertentu dipilih	Sistem menandai data terpilih untuk digunakan pada proses analisis
TC-SC-04	Data Scraper	Navigasi ke halaman lain dan kembali ke data scraper	–	Sistem menampilkan data scraper secara konsisten tanpa kehilangan data

8. Chatbot

Tabel 3.9 Skenario Pengujian Chatbot

ID	Fitur	Skenario Pengujian	Data Uji	Hasil yang Diharapkan
TC-CB-01	Chatbot	Pengguna membuka fitur chatbot	–	Sistem menampilkan antarmuka chatbot dan siap menerima input pengguna
TC-CB-02	Chatbot	Pengguna mengirimkan pertanyaan melalui chatbot	Pertanyaan teks valid	Sistem menampilkan respons chatbot sesuai dengan pertanyaan yang diberikan
TC-CB-03	Chatbot	Pengguna mengirimkan input kosong	Input kosong	Sistem menampilkan pesan validasi atau respons bahwa input tidak valid
TC-CB-04	Chatbot	Pengguna mengirimkan pertanyaan di luar konteks sistem	Pertanyaan tidak relevan	Sistem tetap memberikan respons atau pesan informasi yang sesuai
TC-CB-05	Chatbot	Pengguna melakukan interaksi berulang dengan chatbot	Beberapa pertanyaan berurutan	Sistem mampu menampilkan respons secara konsisten pada setiap interaksi

9. Logout

Tabel 3.10 Skenario Pengujian Logout

ID	Fitur	Skenario Pengujian	Data Uji	Hasil yang Diharapkan
TC-LO-01	Logout	Pengguna melakukan logout dari sistem	–	Sistem mengakhiri sesi pengguna dan mengarahkan ke halaman login
TC-LO-02	Logout	Pengguna mencoba mengakses halaman dashboard setelah logout	–	Sistem menolak akses dan mengarahkan kembali ke halaman login
TC-LO-03	Logout	Pengguna menutup sesi dan membuka ulang aplikasi	–	Sistem meminta pengguna untuk login kembali

3.2.6 Operation

Tahap *operation* merupakan tahapan di mana sistem frontend dashboard analisis sentimen dijalankan dan digunakan dalam lingkungan operasional terbatas untuk memastikan seluruh fungsi berjalan sesuai dengan tujuan perancangan. Pada tahap ini, sistem digunakan untuk menampilkan data hasil analisis sentimen yang diperoleh dari backend, sehingga dapat diamati kestabilan aplikasi, konsistensi penyajian data, serta respons antarmuka pengguna. Tahap operation bertujuan untuk memastikan bahwa aplikasi frontend yang telah dikembangkan dapat digunakan secara fungsional sebagai dashboard analitik, serta mendukung aktivitas pemantauan informasi sentimen oleh pengguna tanpa kendala utama. Tahapan ini dapat menggunakan layanan shared hosting maupun *Virtual Private Server (VPS)* yang telah dikonfigurasi untuk menjalankan dan mempublikasikan website.

3.2.7 *Maintenance*

Tahap *maintenance* dilakukan untuk menjaga kinerja dan stabilitas sistem setelah digunakan pada tahap *operation*. Aktivitas pemeliharaan pada tahap ini mencakup perbaikan kesalahan minor yang ditemukan selama penggunaan sistem, penyesuaian teknis terhadap perubahan kebutuhan atau data, serta penyempurnaan mekanisme pengelolaan data frontend agar tetap berjalan secara optimal. Tahap *maintenance* juga memungkinkan dilakukannya penyesuaian kecil pada struktur komponen atau mekanisme sinkronisasi data tanpa mengubah arsitektur utama sistem, sehingga sistem tetap selaras dengan tujuan penelitian dan dapat digunakan secara berkelanjutan selama periode penelitian.

3.2.8 *Evaluation*

Teknik evaluasi pada penelitian ini dilakukan untuk menilai kesesuaian perilaku sistem frontend terhadap rancangan arsitektur dan Client Data Layer yang telah ditetapkan. Evaluasi dilakukan dengan pendekatan kualitatif-deskriptif, yaitu melalui observasi terhadap perilaku sistem selama proses pengujian berlangsung tanpa melibatkan pengukuran numerik performa secara detail. Pendekatan ini dipilih karena fokus penelitian diarahkan pada perilaku pengelolaan data dan konsistensi tampilan sistem, bukan pada pengujian efisiensi algoritma atau kinerja backend. Sumber data evaluasi diperoleh dari hasil observasi terhadap tampilan dashboard serta perilaku pengelolaan data pada sisi frontend. Observasi dilakukan terhadap bagaimana data ditampilkan pada berbagai komponen antarmuka, bagaimana sistem merespons pembaruan data, serta bagaimana konsistensi data terjaga ketika komponen yang berbeda menggunakan sumber data yang sama. Selain itu, evaluasi juga dilakukan dengan mengamati log permintaan data untuk memastikan bahwa mekanisme pengelolaan data berjalan sesuai dengan rancangan. Sebagai alat bantu evaluasi, penelitian ini memanfaatkan fitur observasi yang disediakan oleh TanStack Query Devtools. Alat bantu ini digunakan untuk memantau status pengambilan data, mekanisme penyimpanan sementara (caching), serta proses sinkronisasi data antar-komponen selama skenario pengujian dijalankan. Penggunaan alat bantu ini bertujuan untuk mendukung proses observasi perilaku sistem secara lebih terstruktur, tanpa bergantung pada detail implementasi kode program. Indikator evaluasi dalam penelitian ini meliputi beberapa aspek utama, yaitu konsistensi data antar-komponen antarmuka, perilaku mekanisme caching dalam mengendalikan permintaan data berulang, serta kemampuan sistem dalam menyinkronkan pembaruan data sesuai dengan kondisi yang terjadi pada sisi backend. Selain itu, evaluasi juga dilakukan untuk memastikan bahwa perilaku sistem frontend telah sesuai dengan perancangan Client Data Layer dan arsitektur frontend

yang direncanakan pada tahap perancangan sistem. Sistem frontend dinilai berhasil apabila seluruh indikator evaluasi tersebut terpenuhi dan tidak ditemukan ketidaksesuaian perilaku sistem selama skenario pengujian dijalankan.

BAB 4

HASIL DAN PEMBAHASAN

4.1 Hasil

4.1.1 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.

4.1.2 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.

4.1.3 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 3.2.

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 pemutaran 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.

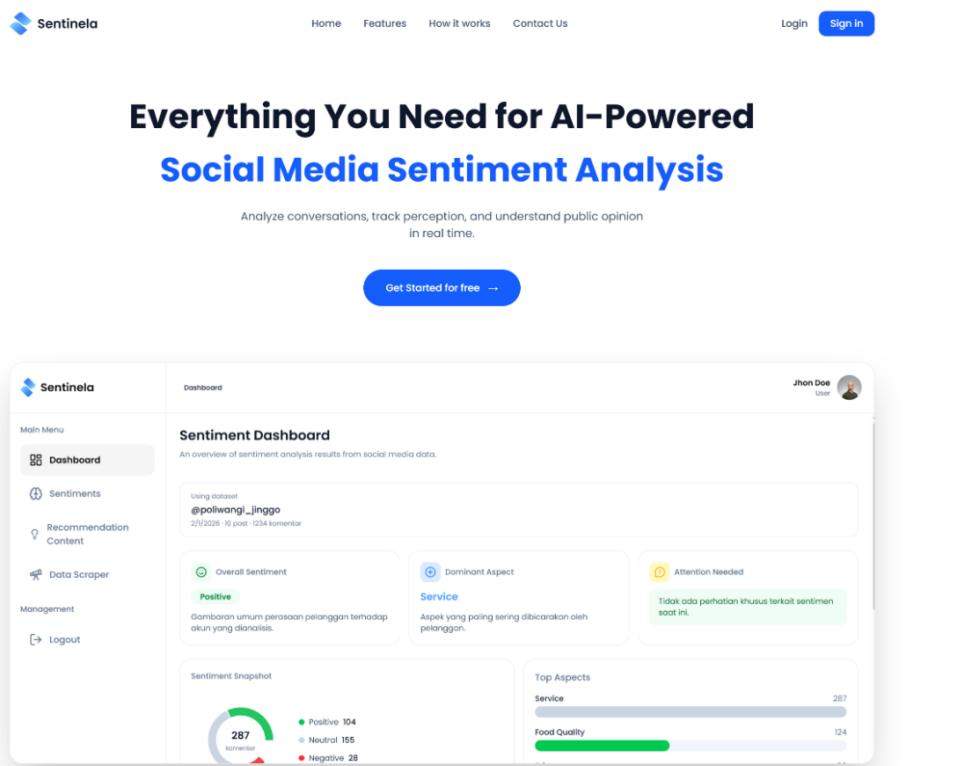
Secara keseluruhan, penerapan *Client Data Layer* menggunakan TanStack Query menghasilkan perilaku pengelolaan data frontend yang lebih terstruktur, konsisten, dan efisien pada tahap implementasi sistem. Lapisan ini mendukung kebutuhan aplikasi dashboard analisis sentimen yang bersifat data-driven, di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

Desain Antarmuka

Desain antarmuka pada sisi frontend dashboard analisis sentimen direalisasikan berdasarkan rancangan wireframe yang telah disusun pada tahap perancangan. Seluruh antarmuka yang diimplementasikan mencakup halaman landing page, login, register, dashboard, analisis sentimen, rekomendasi konten, dan halaman scraper. Implementasi desain antarmuka bertujuan untuk memastikan bahwa setiap fitur sistem dapat diakses dan digunakan sesuai dengan alur yang telah dirancang. Setiap halaman dirancang untuk menyajikan

informasi yang relevan dengan konteks penggunaannya. Adapun hasil realisasi dari wireframe dapat dilihat pada Gambar 4.1 sampai Gambar 4.7.

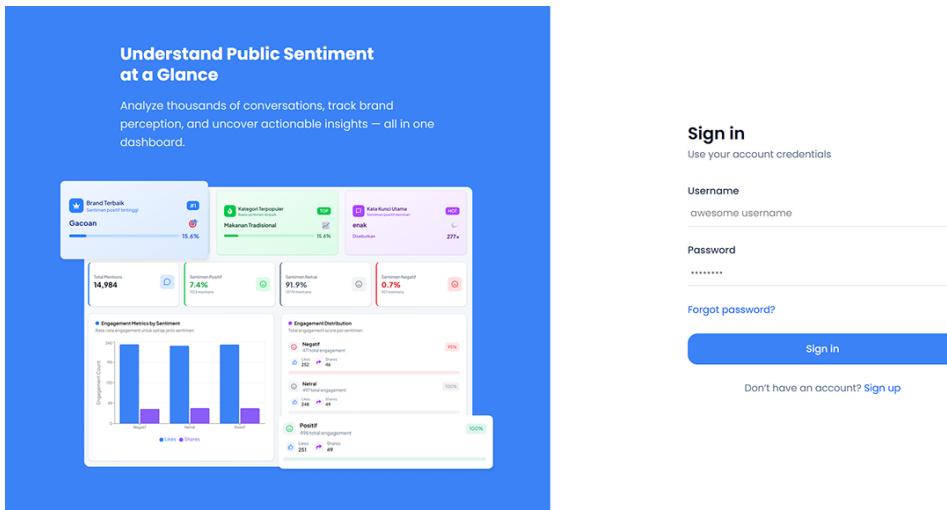
1. Landing Page



Gambar 4.1 Desain Halaman Landing Page

Gambar 4.1 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.

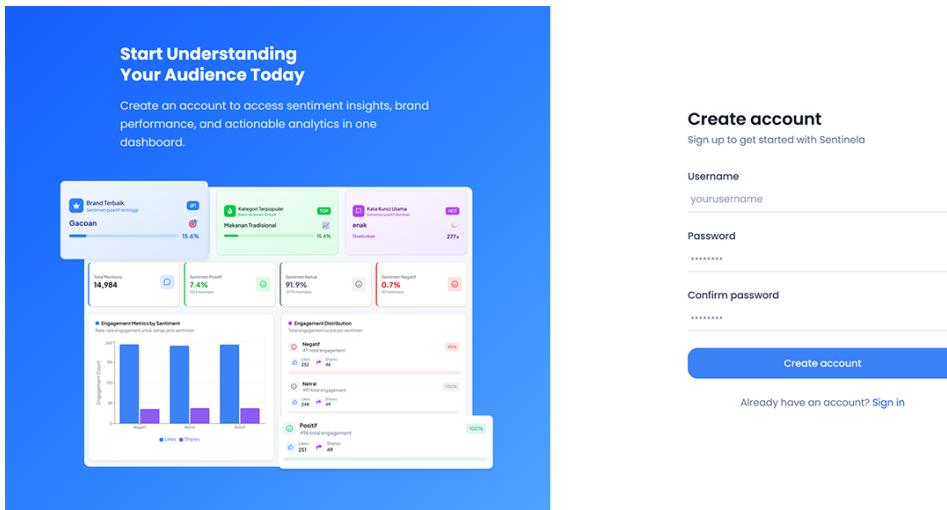
2. Login



Gambar 4.2 Desain Halaman Login

Gambar 4.2 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.

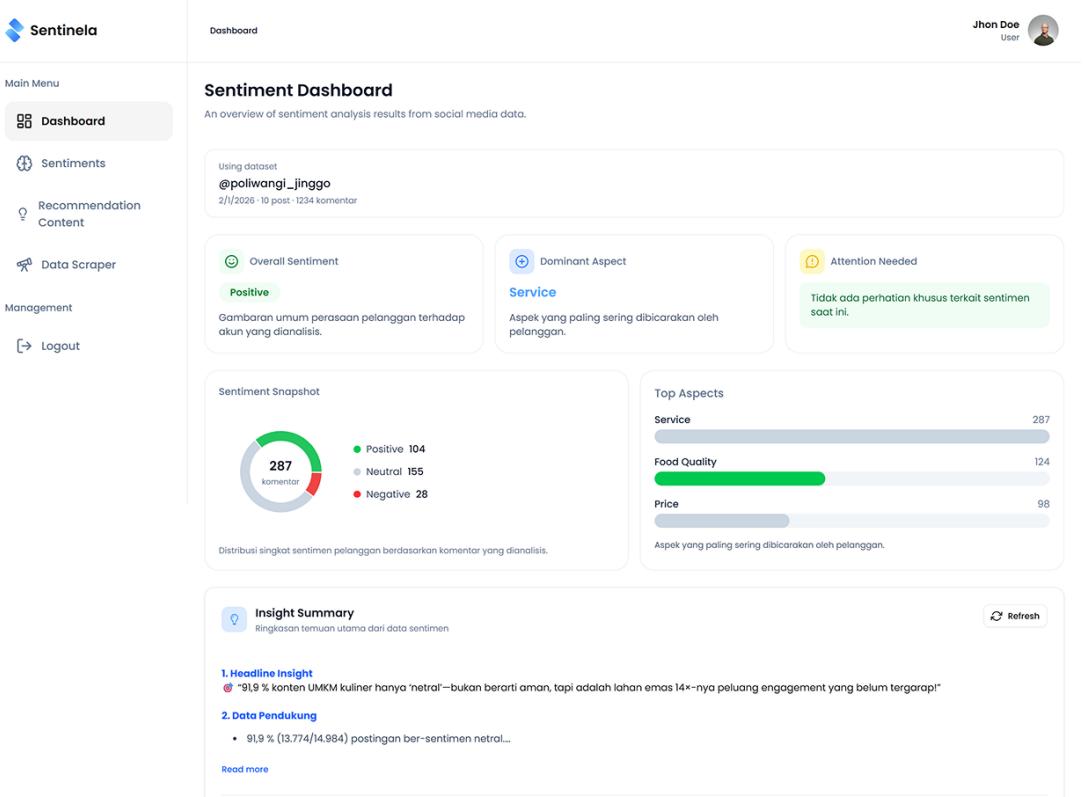
3. Register



Gambar 4.3 Desain Halaman Register

Gambar 4.3 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.

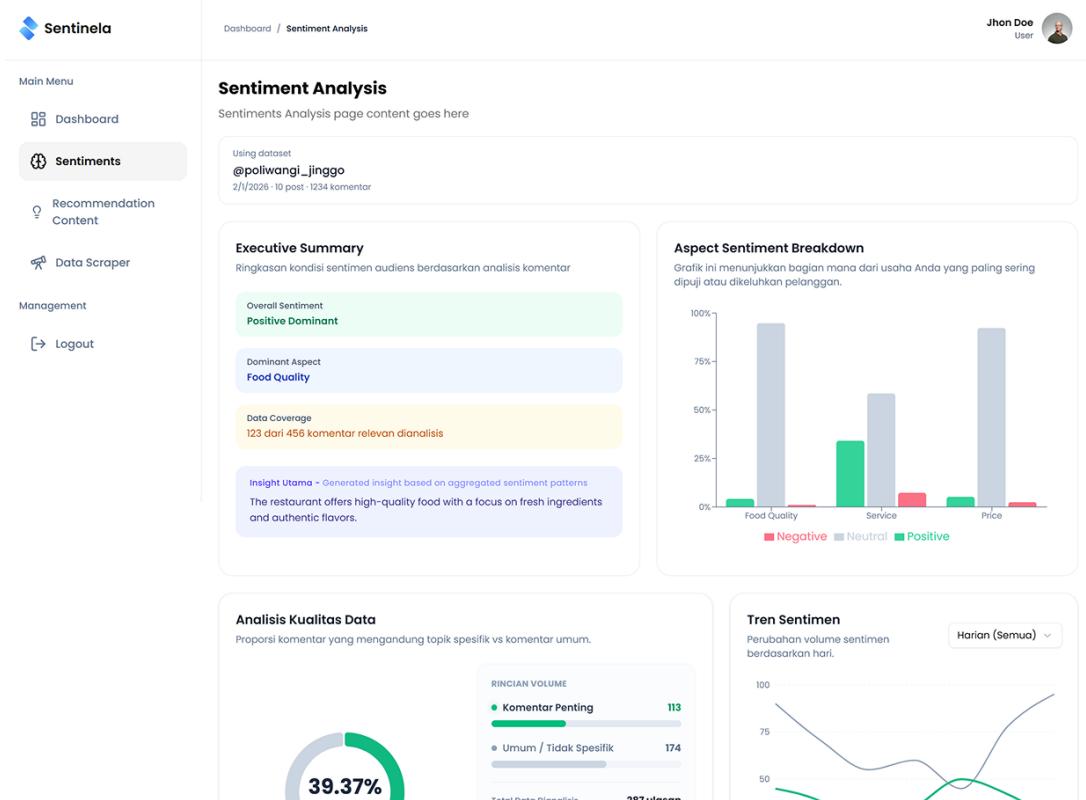
4. Dashboard



Gambar 4.4 Desain Halaman Dashboard

Gambar 4.4 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.

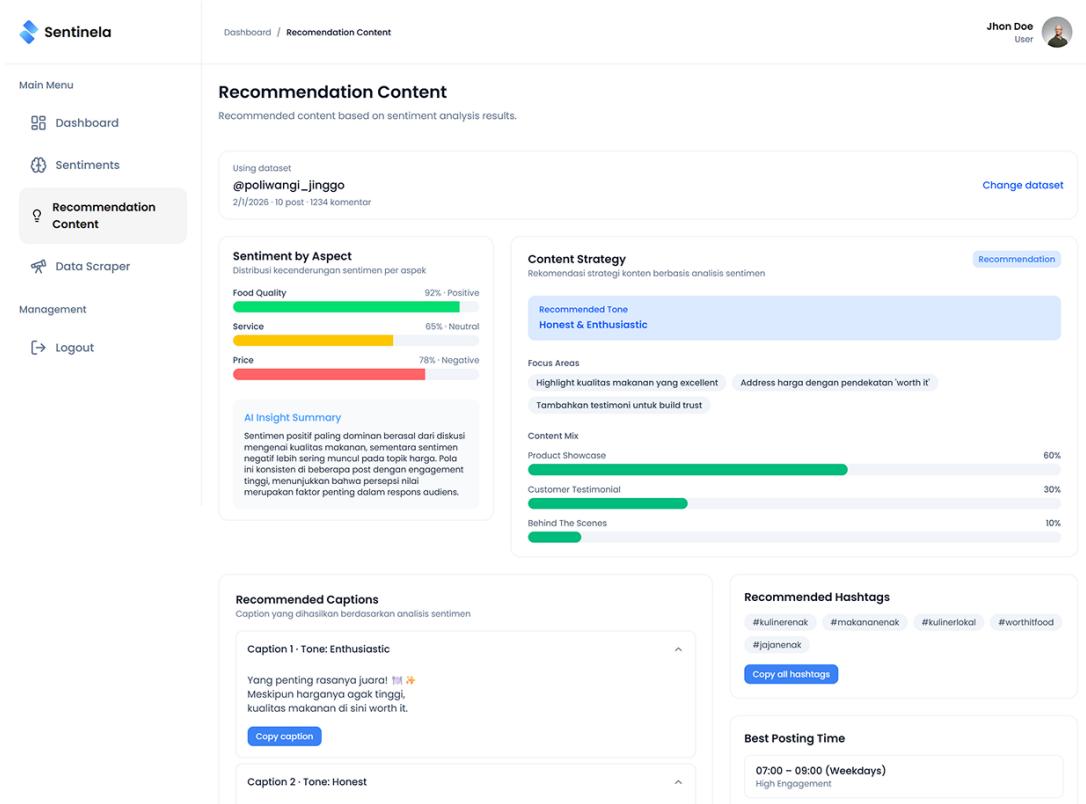
5. Sentiment



Gambar 4.5 Desain Halaman Sentiment

Gambar 4.5 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.

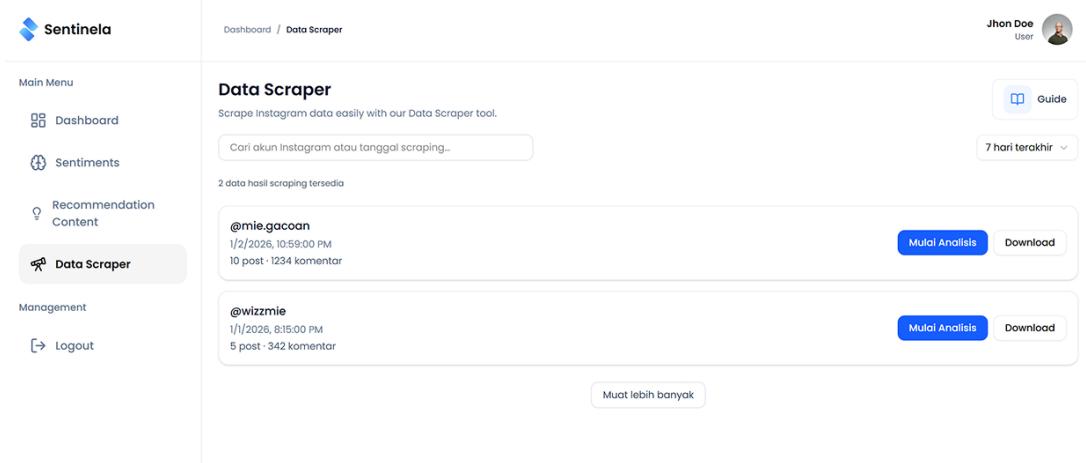
6. Recommendation Content



Gambar 4.6 Desain Halaman Recommendation Content

Gambar 4.6 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.

7. Scraper



Gambar 4.7 Desain Halaman Scraper

Gambar 4.7 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.

4.1.4 *Coding/Implementation*

Landing Page

Authentication

Dashboard

Scraper

Sentiment

Recomendation Content

4.1.5 *Testing*

4.1.6 *Operation*

Pada tahap *operation*, sistem frontend dashboard analisis sentimen telah dijalankan pada lingkungan server berbasis container dan dapat diakses melalui peramban web. Sistem menggunakan runtime JavaScript modern untuk menjalankan aplikasi frontend, sehingga fitur-fitur utama yang telah diimplementasikan dapat digunakan secara fungsional sesuai dengan kebutuhan pengembangan.

Berdasarkan kondisi sistem saat ini, proses operasional aplikasi menunjukkan bahwa alur penggunaan utama dapat dijalankan sesuai dengan perancangan. Meskipun pengembangan sistem masih berada pada tahap lanjutan, implementasi sistem frontend yang telah dilakukan memungkinkan pengguna mengakses dan memanfaatkan fungsi utama sistem sebagai bagian dari proses analisis sentimen.

4.1.7 *Maintenance*

Pada tahap *maintenance*, sistem frontend diimplementasikan dengan struktur yang mendukung proses pemeliharaan dan pengembangan lanjutan. Pemisahan antara komponen antarmuka, logika pengelolaan data, serta definisi struktur data dan aturan validasi memungkinkan perubahan atau perbaikan dilakukan pada satu bagian tanpa memengaruhi keseluruhan sistem.

Struktur frontend yang modular mempermudah proses penyesuaian fitur, perbaikan kesalahan, serta pengembangan tambahan pada tahap selanjutnya. Dengan pendekatan ini,

sistem memiliki kesiapan untuk dilakukan pemeliharaan secara berkelanjutan seiring dengan penyempurnaan proses pengembangan dan pengujian.

4.1.8 *Evaluation*

Evaluasi dilakukan terhadap hasil implementasi *Client Data Layer* pada sistem frontend dengan mengamati perilaku pengelolaan data saat aplikasi dijalankan. Pengamatan dilakukan menggunakan TanStack DevTools sebagai alat bantu untuk melihat status *query*, pemanfaatan cache, serta distribusi data ke komponen antarmuka.

Berdasarkan hasil observasi, data pada sistem frontend dikelola secara terpusat melalui mekanisme *Client Data Layer*. Satu sumber data digunakan bersama oleh beberapa komponen tanpa memerlukan pengelolaan state secara manual pada masing-masing komponen. Selain itu, data yang telah diperoleh tersimpan pada cache dan dapat digunakan kembali selama masih relevan dengan kebutuhan sistem.

Hasil evaluasi ini menunjukkan bahwa penerapan *Client Data Layer* menggunakan TanStack Query telah sesuai dengan tujuan penelitian, yaitu mendukung pengelolaan data frontend yang konsisten dan terstruktur. Evaluasi ini bersifat kualitatif dan dilakukan berdasarkan kondisi implementasi sistem pada tahap pengembangan saat ini.

4.2 Pembahasan

Analisis kinerja sistem dilakukan untuk mengevaluasi efektivitas dan efisiensi sistem dalam mencapai tujuan yang telah ditetapkan. Bagian ini berfokus pada pengukuran dan analisis terhadap parameter-parameter utama yang mencerminkan kualitas kinerja sistem.

BAB 5

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Bagian kesimpulan menyajikan ringkasan dari temuan dan hasil yang diperoleh selama pelaksanaan proyek. Kesimpulan ini menjawab tujuan proyek dan masalah yang telah diidentifikasi di awal laporan, serta mengonfirmasi pencapaian yang telah diraih berdasarkan hasil implementasi dan pengujian. Kesimpulan harus ditarik secara objektif, didasarkan pada data dan hasil yang telah diperoleh, serta tidak memasukkan opini atau asumsi yang tidak didukung oleh hasil pengujian.

Kesimpulan harus dibuat dengan singkat dan jelas, mencakup poin-poin utama yang berhasil dicapai dalam proyek, seperti:

- Capaian utama yang menunjukkan bahwa proyek berhasil memenuhi spesifikasi yang ditetapkan
- Efektivitas sistem dalam menjalankan fungsinya berdasarkan hasil pengujian
- Kesesuaian hasil proyek dengan teori dan standar yang telah diuraikan sebelumnya

Selain itu, kesimpulan juga membahas keterkaitan dengan hasil-hasil penelitian atau proyek serupa yang telah dilakukan sebelumnya, untuk menunjukkan kontribusi dan relevansi dari proyek ini dalam konteks yang lebih luas. Bagian ini juga bisa mencakup hal-hal baru yang ditemukan selama proyek yang dapat memberikan kontribusi positif dalam pengembangan teknologi atau aplikasi di masa mendatang.

Secara keseluruhan, kesimpulan harus memberikan gambaran yang menyeluruh mengenai efektivitas, pencapaian, dan kontribusi proyek terhadap bidang yang diteliti, sekaligus merangkum seluruh hasil dengan ringkas namun komprehensif.

5.2 Saran

Bagian saran menyajikan rekomendasi untuk pengembangan lebih lanjut yang dapat dilakukan berdasarkan temuan dan hasil yang diperoleh dalam proyek ini. Saran diberikan untuk membantu pembaca memahami langkah-langkah tambahan atau perbaikan yang dapat dilakukan untuk menyempurnakan proyek ini atau untuk membuka peluang penelitian atau pengembangan lebih lanjut.

Saran yang diberikan sebaiknya mencakup hal-hal berikut:

- Pengembangan lanjutan pada sistem atau perangkat, seperti peningkatan teknologi atau penambahan fitur yang belum sempat diimplementasikan dalam proyek ini.

- Pengujian lebih lanjut di berbagai kondisi atau lingkungan yang berbeda, untuk memastikan sistem mampu beradaptasi dalam berbagai situasi dan meningkatkan keandalannya.
- Penelitian tambahan untuk menggali aspek-aspek yang belum sepenuhnya terjawab dalam proyek ini atau untuk memvalidasi hasil yang telah diperoleh.
- Pengembangan aplikasi sistem yang lebih luas di bidang lain yang relevan, agar hasil proyek ini dapat memberikan manfaat yang lebih besar di luar bidang awal yang menjadi fokus.

Selain itu, saran juga dapat mencakup rekomendasi untuk mengatasi keterbatasan yang ditemui selama proyek, seperti:

- Penyempurnaan metode atau pendekatan yang digunakan, jika ditemukan kelemahan dalam tahap implementasi atau pengujian
- Peningkatan perangkat keras atau perangkat lunak untuk meningkatkan performa sistem secara keseluruhan
- Pemanfaatan teknologi atau metode baru yang relevan untuk memperbaiki atau menambah kapabilitas sistem

Saran harus dibahas dalam konteks tujuan proyek dan masalah yang diidentifikasi, serta didasarkan pada hasil yang diperoleh. Rekomendasi juga perlu realistik dan dapat diimplementasikan dalam kondisi praktis, agar memberikan panduan yang bermanfaat bagi pengembangan lebih lanjut atau implementasi yang lebih luas.

Secara keseluruhan, saran ini bertujuan untuk memberikan arah bagi pengembangan proyek atau penelitian selanjutnya, sekaligus menunjukkan bagaimana hasil dari proyek ini dapat dioptimalkan dan memberikan kontribusi yang lebih besar dalam bidang yang terkait.

DAFTAR PUSTAKA

- Alviani, N. A., Studi, P., Fakultas, M., & Bangsa, U. B. (2025). Transformasi Digital pada UMKM dalam Meningkatkan Daya Saing Pasar. *Master Manajemen*, 3(1), 134–140.
- Aniley, D., Alemneh, E., & Abeba, G. (2024). Selection of software development life cycle models using machine learning approach. *International Journal of Computer Applications*, 186, 975–8887.
- Fajarini, S., Kurniawati, J., & 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.
- 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.
- Mardhatilah, D., Omar, A., & 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., & 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.
- Putra, F. P. E., Efendi, R. W., Tamam, A. B., & 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. & 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., & 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. & 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., & 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. & Agusman (2025). ANALISIS KESIAPAN UMKM MENGHADAPI EKONOMI DIGITAL. *Smart Jurnal Ilmiah*, IX(1), 1–4.
- 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 A

KODE PROGRAM

Lampiran A.1. Program Pembacaan Sensor Ultrasonic

Lampiran A.2. Program Keseluruhan Proyek Akhir

— Halaman ini sengaja dikosongkan —

LAMPIRAN B

GAMBAR-GAMBAR

Lampiran B.1. Foto Aktivitas Kegiatan Proyek Akhir



Lampiran B.2. Foto Produk Proyek Akhir

