RANCANG BANGUN APLIKASI WEB GUNA PREDIKSI POLIMER DENGAN INTEGRASI *AGENTIC AI*

TUGAS AKHIR

Diajukan oleh:

ZAKIUL FATA
NIM. 220705051

Mahasiswa Fakultas Sains dan Teknologi

Program Studi Teknologi Informasi

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI AR-RANIRY

BANDA ACEH

2026 M/1448 H

LEMBAR PERSETUJUAN

RANCANG BANGUN APLIKASI WEB GUNA PREDIKSI POLIMER DENGAN INTEGRASI *AGENTIC AI*

TUGAS AKHIR

Diajukan kepada Fakultas Sains dan Teknologi
Universitas Islam Negeri Ar-Raniry Banda Aceh
Sebagai Salah Satu Beban Studi Memperoleh Gelar Sarjana (S1)
Dalam Ilmu Teknologi Informasi

Oleh:
ZAKIUL FATA
NIM. 220705051

Mahasiswa Fakultas Sains dan Teknologi
Program Studi Teknologi Informasi

Disetujui Untuk di Munaqasyahkan Oleh:

Pembimbing I,	Pembimbing II,
Nazaruddin Ahmad, M.T	**Khairan AR, M.Kom**
NIP. 198206052014031002	**NIP. 198607042014031001**

Mengetahui,
Ketua Program Studi Teknologi Informasi

Malahayati, M.T
NIP. 198301272015032003

LEMBAR PENGESAHAN

RANCANG BANGUN APLIKASI WEB GUNA PREDIKSI POLIMER DENGAN INTEGRASI *AGENTIC AI*

TUGAS AKHIR

Telah Diuji Oleh Panitia Ujian Munaqasyah Tugas Akhir
Fakultas Sains dan Teknologi UIN Ar-Raniry Banda Aceh dan Dinyatakan Lulus
Serta Diterima Sebagai Salah Satu Beban Studi Program Sarjana (S1)
Dalam Program Studi Teknologi Informasi

Pada Hari/Tanggal: Rabu, 05 Agustus 2026 M
21 Shafar 1448 H
Di Darussalam, Banda Aceh

Panitia Ujian Munaqasyah Tugas Akhir:

Ketua,	Sekretaris,
Nazaruddin Ahmad, M.T	**Khairan AR, M.Kom**
NIP. 198206052014031002	**NIP. 198607042014031001**
Penguji I,	Penguji II,
Rahmad Ade Akbar, M.Pd	**Dr. Hendri Ahmadian, S.Si., M.I.M**
	NIP. 198301042014031002

Mengetahui:
Dekan Fakultas Sains dan Teknologi
UIN Ar-Raniry Banda Aceh,

Prof. Dr. Ir. Muhammad Dirhamsyah, M.T., IPU
NIP. 196210021988111001

LEMBAR PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama	: Zakiul Fata
Nim	: 220705051
Program Studi	: Teknologi Informasi
Fakultas	: Sains dan Teknologi
Judul	: Rancang Bangun Aplikasi Web Guna Prediksi Polimer Dengan Integrasi *Agentic AI*

Dengan ini menyatakan bahwa dalam penulisan tugas akhir ini, saya:

1. Tidak menggunakan ide orang lain tanpa mampu mengembangkan dan mempertanggungjawabkan;
2. Tidak melakukan plagiasi terhadap naskah karya orang lain;
3. Tidak menggunakan karya orang lain tanpa menyebutkan sumber asli atau tanpa izin pemilik karya;
4. Tidak memanipulasi dan memalsukan data;
5. Mengerjakan sendiri karya ini dan mampu bertanggungjawab atas karya ini.

Bila dikemudian hari ada tuntutan dari pihak lain atas karya saya, dan telah melalui pembuktian yang dapat dipertanggungjawabkan dan ternyata memang ditemukan bukti bahwa saya telah melanggar pernyataan ini, maka saya siap dikenai sanksi berdasarkan aturan yang berlaku di Fakultas Sains dan Teknologi UIN Ar-Raniry Banda Aceh. Demikian pernyataan ini saya buat dengan sesungguhnya dan tanpa paksaan dari pihak manapun.

Banda Aceh, 05 Agustus 2026

Yang Menyatakan

Zakiul Fata

ABSTRAK

Nama	:	Zakiul Fata
NIM	:	220705051
Program Studi	:	Teknologi Informasi
Judul	:	Rancang Bangun Aplikasi Web Guna Prediksi Polimer Dengan Integrasi *Agentic AI*

Tanggal Sidang	:	05 Agustus 2026
Jumlah Halaman	:	124 Halaman
Pembimbing I	:	Nazaruddin Ahmad, M.T
Pembimbing II	:	Khairan AR, M. Kom
Kata Kunci	:	Prediksi Polimer, *Agentic AI*, *Large Language Model*, Aplikasi Web, FastAPI, React.js, *RDKit.*

Penemuan senyawa polimer baru melalui metode konvensional eksperimen laboratorium sering kali memakan waktu lama dan biaya yang tinggi. Meskipun pemanfaatan kecerdasan buatan (AI) generatif menjanjikan akselerasi penemuan, sistem ini rentan terhadap halusinasi data jika tidak didukung oleh validasi struktural kimia yang pasti. Selain itu, sebagian besar alat prediksi *cheminformatics* saat ini masih berbasis antarmuka baris perintah (CLI) yang sulit diakses oleh peneliti non-programer. Penelitian ini bertujuan untuk merancang dan membangun aplikasi web bernama "PolyChem" guna memprediksi properti termal polimer, khususnya nilai *Glass Transition Temperature* (Tg), dengan mengintegrasikan teknologi *Agentic AI*. Penelitian ini menggunakan metode pengembangan perangkat lunak *Prototyping* dengan mengadopsi arsitektur *Client-Server* yang terpisah (*decoupled*). Sisi *frontend* dibangun menggunakan React.js untuk menghasilkan antarmuka yang interaktif, sementara sisi *backend* diorkestrasi menggunakan *framework* FastAPI guna menangani beban komputasi tingkat tinggi, serta Firebase untuk manajemen autentikasi dan basis data. Sistem ini menerapkan arsitektur *Agentic AI* berbasis *neuro-symbolic*, yang memadukan komputasi deterministik RDKit untuk validasi struktural dengan *Large Language Model* (LLM) sebagai agen penalaran ilmiah. Guna menjamin keandalan dan ketersediaan sistem dari limitasi kuota maupun gangguan jaringan, diterapkan arsitektur *multi-*

provider yang memadukan Google Gemini 2.5 Flash sebagai agen utama dan Groq (Llama 3.3) sebagai agen cadangan (*fallback*) dengan mekanisme *fail-fast*. Hasil pengujian fungsionalitas sistem menggunakan metode *Black Box Testing* secara *End-to-End* (E2E) menunjukkan bahwa seluruh fitur aplikasi berjalan secara valid dan tangguh (*robust*). Mekanisme peralihan *multi-provider* terbukti sukses menjaga sistem agar tetap responsif tanpa mengalami kegagalan (*crash*) saat layanan utama terinterupsi. Aplikasi web PolyChem berhasil menerjemahkan komputasi kimia yang kompleks menjadi antarmuka visual yang intuitif, sehingga memungkinkan pengguna untuk melakukan skrining polimer secara *in silico* dengan cepat, mendapatkan prediksi Tg, serta memperoleh justifikasi saintifik yang dapat diandalkan.

ABSTRACT

Name	:	Zakiul Fata
Student Number	:	220705051
Depertement	:	Information Technology
Title	:	Design and Development of a Web Application for Polymer Prediction with Agentic AI Integration

Date	:	05 August 2026
Number Of Pages	:	124 Pages
Supervisor I	:	Nazaruddin Ahmad, M.T
Supervisor II	:	Khairan AR, M. Kom
Keywords	:	Polymer Prediction, Agentic AI, Large Language Models, Web Application, FastAPI, React.js, RDKit.

The discovery of new polymer compounds through conventional laboratory experiments is often time-consuming and highly expensive. Although the utilization of generative artificial intelligence (AI) promises to accelerate discovery, these systems are prone to data hallucinations if not supported by definitive chemical structural validation. Furthermore, most current cheminformatics prediction tools remain Command Line Interface (CLI)-based, making them difficult for non-programmer researchers to access. This research aims to design and develop a web application named "PolyChem" to predict the thermal properties of polymers, specifically the Glass Transition Temperature (Tg) values, by integrating Agentic AI technology. This study employed the Prototyping software development method, adopting a decoupled Client-Server architecture. The frontend was built using React.js to provide an interactive interface, while the backend was orchestrated using the FastAPI framework to handle high computational loads, alongside Firebase for authentication and database management. The system implemented a neuro-symbolic Agentic AI architecture, combining RDKit's deterministic computations for structural validation with Large Language Models (LLMs) as scientific reasoning agents. To ensure system reliability and availability against quota limitations or network disruptions, a multi-provider architecture was implemented, integrating Google Gemini 2.5 Flash as the

primary agent and Groq (Llama 3.3) as the fallback agent with a fail-fast mechanism. The system functionality evaluation results, conducted using End-to End (E2E) Black Box Testing, indicated that all application features operated validly and robustly. The multi-provider switching mechanism successfully maintained system responsiveness without experiencing crashes during primary service interruptions. The PolyChem web application successfully translated complex chemical computations into an intuitive visual interface, thereby enabling users to conduct rapid in silico polymer screening, obtain Tg predictions, and acquire reliable scientific justifications.

KATA PENGANTAR

Assalamu'alaikum Wr. Wb

Alhamdulillahirabbil'alamin, puja dan puji serta syukur kita ucapkan kehadirat Allah SWT, berkat nikmat dan karunia-Nya yang indah yang masih kita rasakan sampai pada saat ini, nikmat berupa iman, Islam, kesehatan, kesempatan, pengetahuan yang tentunya masih banyak lagi nikmat yang tidak dapat dijabarkan di atas seluruh kertas ini.

Dalam kesempatan ini penulis bersyukur kepada Allah SWT, karena berkat Ridho-Nya penulis mampu merampungkan tugas akhir yang berjudul **"Rancang Bangun Aplikasi Web Guna Prediksi Polimer Dengan Integrasi *Agentic AI*"**. Tugas akhir ini disusun sebagai kewajiban penulis guna melengkapi tugas dan syarat untuk menyelesaikan pendidikan Strata-I (S1) Program Studi Teknologi Informasi Fakultas Sains dan Teknologi Universitas Islam Negeri Ar-Raniry Banda Aceh, serta memperoleh gelar Sarjana Komputer.

Dalam proses penyusunan tugas akhir ini, penulis telah mendapatkan banyak bantuan, bimbingan, dan dukungan dari berbagai pihak. Oleh karena itu, dengan segala hormat dan rasa syukur, penulis menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Kedua orang tua tercinta, Ayahanda Abdullah Isa dan Almarhumah Ibunda Ummi Salamah, yang senantiasa menjadi sumber kekuatan terbesar penulis. Terima kasih atas segala pengorbanan, kasih sayang, dan doa yang tak pernah putus. Semoga Allah SWT senantiasa memuliakan dan menjaga Ayahanda, serta mengaruniakan tempat terindah di surga-Nya bagi Almarhumah Ibunda.
2. Bapak Prof. Dr. Ir. M. Dirhamsyah, M.T., IPU selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Ar-Raniry Banda Aceh.
3. Ibu Malahayati, M.T. dan Bapak Khairan AR, M.Kom. selaku Ketua dan Sekretaris Program Studi Teknologi Informasi yang telah memberikan arahan, kebijakan, serta dukungan akademik selama penulis menempuh pendidikan.
4. Bapak Nazaruddin Ahmad, M.T. dan Bapak Khairan AR, M.Kom. selaku Pembimbing I dan Pembimbing II Tugas Akhir, atas segala ilmu, bimbingan

teknis, dan kesabaran dalam mengarahkan penulis hingga aplikasi ini dapat diselesaikan dengan baik.

5. Ibu Cut Ida Rahmadiana, S.Si., yang telah membantu penulis dalam pengurusan administrasi dan surat-menyurat.

6. Saudara kandung Kakak dan Abang tercinta, atas segala bentuk dukungan, doa, dan motivasi yang terus mengalir. Kepedulian, bantuan, dan kebersamaan kalian menjadi energi penyemangat tersendiri bagi penulis dalam menyelesaikan tugas akhir ini.

7. Kepada Hilda Sabrina Hashar selaku *partner* penulis, terima kasih atas kehadiran, waktu, dan segala dukungan yang diberikan. Terima kasih telah menjadi pendengar setia, *support system* terbaik, serta memberikan banyak saran dan bantuan selama penulisan karya ilmiah ini. Terima kasih karena selalu membersamai dan menerima segala kekurangan penulis apa adanya.

8. Rekan satu tim dalam pengembangan PolyChem, Dikky Julianto, Tiara Diansyah Putri dan Loista Amanda Noviar. Terima kasih atas kolaborasi yang solid, kerja keras tanpa lelah, serta semangat yang tak pernah surut dalam mewujudkan aplikasi ini.

9. Kepada sahabat-sahabat terbaik penulis: M. Daffa Al Ma'arif, Hilda Sabrina Hashar, Farras Nabil Rivanza, Arif Maulana, dan Arif Maulana Insaf, serta keluarga besar grup Revolusi Warkop dan AHK. Terima kasih atas solidaritas, canda tawa, dan dukungan semangat yang tidak pernah putus. Kehadiran kalian telah menjadi pelipur penat dan motivasi besar bagi penulis dalam menyelesaikan karya ilmiah ini.

Peneliti menyadari bahwa Tugas Akhir ini masih jauh dari kata sempurna. Oleh karena itu, kritik dan saran yang membangun selalu peneliti harapkan, demi penyusunan Tugas Akhir yang baik. Peneliti berharap semoga tulisan ini dapat bermanfaat bagi perkembangan ilmu pengetahuan.

Banda Aceh, Juli 2026

Penulis

Zakiul Fata

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Industri material dan kimia saat ini menghadapi tantangan besar dalam akselerasi penemuan senyawa polimer baru. Metode konvensional yang mengandalkan eksperimen laboratorium *(wet lab)* sering kali memakan waktu lama, biaya tinggi, dan melibatkan proses *trial-and-error* yang tidak efisien. Dalam kurun waktu terakhir, paradigma penelitian mulai bergeser ke arah pemanfaatan *Large Language Model (LLM)* dan *Agentic AI*. Teknologi *Agentic AI* menawarkan kemampuan yang melampaui model generatif biasa, di mana sistem tidak hanya memproses teks, tetapi juga mampu merencanakan, mengambil keputusan, dan mengeksekusi tugas secara otonom melalui interaksi dengan layanan eksternal untuk menyelesaikan alur kerja yang kompleks (Bandi et al., 2025).

Meskipun potensi *Agentic AI* sangat besar, penerapannya dalam aplikasi nyata menuntut infrastruktur perangkat lunak yang matang. Integrasi kecerdasan buatan ke dalam aplikasi web modern tidak lagi dipandang sebagai komponen tambahan, melainkan harus tertanam langsung ke dalam arsitektur sistem untuk menjamin kelancaran pertukaran data (Wang & Li, 2024). Masalah utamanya adalah sebagian besar alat prediksi kimia berbasis AI saat ini masih berjalan pada antarmuka baris perintah *(Command Line Interface)* yang sulit diakses oleh peneliti *non-programer*. Oleh karena itu, diperlukan antarmuka visual yang intuitif. Penelitian menunjukkan bahwa desain antarmuka yang responsif berperan signifikan dalam meningkatkan produktivitas serta kepuasan pengguna saat berinteraksi dengan sistem berbasis AI (Buendía-García & Piris-Ruano, 2025).

Untuk mengatasi tantangan integrasi dan aksesibilitas tersebut, penelitian ini mengusulkan pengembangan aplikasi web "PolyChem" dengan arsitektur *Client-Server* yang terpisah. Pemilihan teknologi *backend* menjadi sangat krusial untuk menangani beban komputasi AI. Kerangka kerja FastAPI dipilih karena terbukti mampu menangani beban kerja tinggi, memiliki performa yang setara dengan NodeJS atau Go, serta mempermudah integrasi layanan berbasis API

dengan ekosistem Python (Safitri & Harkespan, 2024). Penggunaan FastAPI memungkinkan orkestrasi antara logika *Agentic AI* dan validasi kimia berjalan efisien tanpa membebani sisi antarmuka. Sementara itu, di sisi *frontend*, penggunaan React.js diterapkan untuk membangun antarmuka yang dinamis. Pendekatan arsitektur berbasis API *(API-driven)* ini memungkinkan komponen sistem berkembang secara independen tanpa mengganggu keseluruhan sistem, sebuah praktik yang sangat disarankan dalam pengembangan aplikasi web modern (Richards & Ford, 2020).

Berdasarkan urgensi tersebut, penelitian ini bertujuan merancang dan membangun aplikasi web prediksi polimer (PolyChem) yang mengintegrasikan *Agentic AI* ke dalam sebuah sistem yang mudah digunakan oleh peneliti. Aplikasi ini diharapkan menjadi solusi perangkat lunak yang mampu meningkatkan aksesibilitas terhadap teknologi prediksi material, sehingga proses *in silico screening* dapat dilakukan secara lebih efektif dan efisien. Fokus utama sistem adalah melakukan prediksi properti termal polimer berupa *Glass Transition Temperature* (Tg) berdasarkan representasi struktur molekul dalam format SMILES. Parameter Tg dipilih karena merupakan salah satu indikator penting dalam menentukan performa material polimer, khususnya terkait fleksibilitas dan stabilitas termalnya.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan oleh penulis maka rumusan masalah dari penelitian ini adalah sebagai berikut:

1. Bagaimana merancang dan membangun aplikasi web untuk prediksi properti polimer berbasis struktur molekul?
2. Bagaimana mengintegrasikan model prediksi polimer ke dalam arsitektur sistem berbasis *Agentic AI*?

1.3 Tujuan Penelitian

Berdasarkan latar belakang dan juga rumusan masalah yang telah diuraikan oleh penulis maka tujuan dari penelitian ini adalah sebagai berikut:

1. Merancang dan membangun aplikasi web untuk prediksi properti polimer berbasis struktur molekul.
2. Mengintegrasikan model prediksi polimer ke dalam arsitektur aplikasi web berbasis *Agentic AI*.

1.4 Manfaat Penelitian

Berdasarkan latar belakang, rumusan masalah dan juga tujuan penelitian yang telah diuraikan oleh penulis maka manfaat dari penelitian ini yaitu :

1. Menjadi referensi akademik dalam pengembangan aplikasi berbasis Artificial Intelligence khususnya pada integrasi sistem AI dalam bidang cheminformatics.
2. Membantu pengguna dalam melakukan prediksi awal properti polimer melalui aplikasi web yang lebih mudah digunakan dibandingkan tools komputasi yang bersifat teknis.
3. Menjadi contoh implementasi integrasi *Agentic AI* dalam pengembangan aplikasi berbasis web.
4. Menunjukkan penerapan arsitektur frontend dan *backend* dalam sistem AI berbasis web.
5. Menjadi dasar pengembangan penelitian selanjutnya khususnya pada pengembangan fitur dan peningkatan kemampuan sistem prediksi polimer.

1.5 Batasan Masalah

Untuk memastikan kesesuaian, penulis memutuskan untuk mempersempit cakupan masalah yang akan dibahas dalam penelitian ini dengan menetapkan batasan sebagai berikut:

1. Penelitian ini berfokus pada rancang bangun aplikasi web prediksi polimer, bukan pada pengembangan model machine learning baru.
2. Model prediksi yang digunakan merupakan model yang sudah tersedia dan hanya diintegrasikan ke dalam sistem.
3. Prediksi yang dilakukan dibatasi pada properti termal polimer yaitu Glass Transition Temperature (Tg).
4. Input sistem dibatasi pada struktur molekul dalam format SMILES.
5. Penelitian ini tidak membahas sintesis kimia polimer secara eksperimental (wet lab).
6. Validasi difokuskan pada pengujian fungsionalitas sistem, bukan pada evaluasi performa model AI.

BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Guna memetakan posisi kebaruan (*state of the art*) penelitian ini, dilakukan tinjauan komprehensif terhadap literatur terkini yang mencakup tiga domain utama: penerapan kecerdasan buatan dalam sintesis material, evolusi sistem otonom (*Agentic AI*), serta standar infrastruktur *backend* untuk komputasi berkinerja tinggi. Pemetaan ini bertujuan untuk mengidentifikasi metodologi terbaik yang telah teruji secara empiris sekaligus menemukan celah penelitian (*research gap*) yang belum tersentuh. Rangkuman penelitian-penelitian kunci yang menjadi landasan pengembangan aplikasi web PolyChem disajikan dalam Tabel 2.1.

Tabel 2. 1 Penelitian Terdahulu

Judul dan Peneliti	Metode Penelitian	Hasil Penelitian
Generative BigSMILES: an extension for polymer informatics, computer simulations & ML/AI (Schneider et al., 2024)	Pengembangan Model & Studi Komputasi yang mengintegrasikan notasi kimia (SMILES) dengan algoritma *Machine Learning* untuk merepresentasikan struktur polimer stokastik.	Menunjukkan bahwa penggunaan representasi berbasis teks (SMILES) yang ditingkatkan sangat efektif untuk mendukung alur kerja AI generatif, mempercepat prediksi properti material, dan memfasilitasi desain polimer secara *in silico*.

Performance Analysis of FastAPI Framework on Lost Circulation Handling Management Application in Oil Well Drilling (Suryotomo et al., 2024)	Experimental Load Testing menggunakan JMeter untuk mengukur latensi dan *throughput* server FastAPI saat menerima request yang memicu proses inferensi model *Machine Learning* (klasifikasi & prediksi).	Membuktikan bahwa FastAPI sangat efisien untuk menangani tugas komputasi berat (*compute-heavy tasks*) seperti inferensi AI. Hasil menunjukkan FastAPI tetap stabil dengan *response time* cepat bahkan saat menangani ratusan *concurrent users*, memvalidasi keandalannya sebagai *backend* layanan AI.
The Rise of Agentic AI: A Review of Definitions, Frameworks, Architectures, Applications, Evaluation Metrics, and Challenges (Bandi et al., 2025)	Systematic Literature Review (SLR) mengenai taksonomi, arsitektur, dan metrik evaluasi sistem kecerdasan buatan otonom.	Menyusun kerangka kerja standar untuk *Agentic AI*, mengklasifikasikannya sebagai sistem yang mampu melakukan perencanaan (*planning*) dan eksekusi alat (*tool-use*) secara mandiri, berbeda dengan model generatif pasif.
Augmenting Large Language Models with Chemistry Tools (Bran et al., 2024)	Eksperimen Empiris *(Empirical Experiment)* yang membandingkan akurasi prediksi antara LLM standar (GPT-4) dengan sistem *Neuro-Symbolic* yang dilengkapi alat kimia (ChemCrow).	Membuktikan bahwa integrasi LLM dengan alat eksternal (seperti RDKit) memitigasi halusinasi dan meningkatkan kemampuan penyelesaian masalah kimia kompleks yang tidak dapat diselesaikan oleh LLM biasa.

14 Examples of How LLMs Can Transform Materials Science and Chemistry: A Qualitative Perspective (Jablonka et al., 2023)	Studi Kasus Kualitatif *(Qualitative Case Studies)* terhadap berbagai skenario aplikasi LLM dalam ekstraksi properti material.	Mendemonstrasikan bahwa LLM memiliki kapabilitas penalaran (*reasoning*) yang kuat untuk memahami konteks kimia dan mengekstraksi informasi terstruktur dari data material yang tidak terstruktur.

Berdasarkan sintesis pada Tabel 2.1, terlihat adanya konvergensi teknologi yang signifikan dalam domain kimia komputasi. Schneider et al. (2024) dan Jablonka et al. (2023) secara fundamental memvalidasi bahwa integrasi AI dan informatika polimer mampu menggantikan metode manual dalam memprediksi dan mendesain struktur material baru. Namun, literatur terbaru dari Bandi et al. (2025) dan Bran et al. (2024) menegaskan bahwa model AI generatif semata tidak cukup, diperlukan pergeseran menuju arsitektur *Agentic AI* yang bersifat *neuro-symbolic*. Bran et al. (2024) membuktikan secara empiris bahwa menggabungkan LLM dengan alat eksternal (*tools*) adalah metode paling efektif untuk memitigasi halusinasi dan meningkatkan akurasi kalkulasi kimia.

Di sisi lain, dari aspek implementasi teknis, Suryotomo et al. (2024) memberikan bukti kuantitatif bahwa penggunaan FastAPI sebagai *backend* sangat krusial untuk menangani beban komputasi inferensi model *Machine Learning* yang berat, menjamin stabilitas sistem di bawah *concurrency* tinggi. Mengacu pada paparan tersebut, penelitian ini dirancang untuk mengisi kesenjangan (*gap*) implementasi: di mana konsep *Agentic AI* yang divalidasi oleh Bran et al. (2024) akan diorkestrasi menggunakan infrastruktur berkinerja tinggi berbasis FastAPI (Suryotomo et al., 2024), dan dikemas dalam aplikasi web interaktif. Pendekatan ini menawarkan solusi yang tidak hanya akurat secara ilmiah, tetapi juga responsif dan mudah diakses oleh pengguna akhir.

2.2 Landasan Teori

2.2.1 Kecerdasan Buatan

Kecerdasan buatan atau *Artificial Intelligence* (AI) merupakan cabang ilmu komputer yang berfokus pada pengembangan sistem yang mampu meniru kemampuan kognitif manusia, seperti penalaran, pembelajaran, pengambilan keputusan, dan pemecahan masalah (Sarker, 2022). Dalam perkembangannya, AI telah banyak diterapkan pada berbagai bidang, termasuk sistem informasi, pengolahan data, dan aplikasi berbasis *web* untuk meningkatkan efisiensi serta kualitas layanan digital.

Pada periode 2022–2025, pemanfaatan AI mengalami peningkatan signifikan seiring dengan kemajuan teknologi komputasi dan ketersediaan data dalam jumlah besar (*big data*). AI tidak hanya digunakan sebagai alat analisis statis, tetapi juga diintegrasikan langsung ke dalam arsitektur sistem aplikasi untuk mendukung proses otomatisasi dan pengambilan keputusan secara cerdas (Hou et al., 2024). Dalam konteks penelitian ini, AI dimanfaatkan sebagai layanan inti (*core service*) yang diintegrasikan melalui API, di mana fokus penelitian menitikberatkan pada aspek rekayasa perangkat lunak (*software engineering*) untuk mengorkestrasi layanan tersebut agar dapat diakses oleh pengguna akhir.

2.2.2 *Large Language Model* (LLM)

Large Language Model (LLM) merupakan model kecerdasan buatan yang dilatih menggunakan korpus data teks dalam skala besar untuk memahami dan menghasilkan bahasa alami. LLM mampu memproses instruksi berbasis teks, memahami konteks linguistik, serta menghasilkan respons yang relevan terhadap masukan pengguna (Jablonka et al., 2023).

LLM banyak dimanfaatkan dalam pengembangan aplikasi berbasis *web*, khususnya untuk mendukung interaksi pengguna dan pemrosesan informasi berbasis bahasa. Dalam konteks penelitian ini, LLM digunakan sebagai bagian dari sistem prediksi yang telah tersedia dan diintegrasikan ke dalam aplikasi *web* PolyChem melalui layanan API, tanpa dilakukan proses pelatihan atau pengembangan model dari awal.

2.2.3 *Agentic AI*

Agentic AI merupakan paradigma baru dalam sistem kecerdasan buatan yang menekankan pada kemampuan agen untuk bertindak secara otonom (*autonomous*) dalam menjalankan tugas tertentu. Berbeda dengan model AI konvensional yang bersifat pasif, sistem *Agentic AI* mampu melakukan perencanaan (*planning*), pengambilan keputusan (*decision making*), serta berinteraksi secara aktif dengan berbagai alat (*tools*) dan layanan eksternal untuk mencapai tujuan yang telah ditetapkan (Bandi et al., 2025).

Secara umum, mekanisme kerja *Agentic AI* melibatkan beberapa tahapan utama, yaitu *perception*, *reasoning*, *action*, dan *evaluation*. Pada tahap *perception*, sistem menerima dan memahami input yang diberikan oleh pengguna. Selanjutnya, pada tahap *reasoning*, sistem melakukan analisis serta penalaran terhadap data yang diterima. Tahap *action* merupakan proses di mana sistem menjalankan aksi tertentu, seperti pemanggilan fungsi, penggunaan alat eksternal, atau pemrosesan lanjutan. Terakhir, pada tahap *evaluation*, sistem melakukan evaluasi terhadap hasil yang dihasilkan untuk memastikan kesesuaian dengan tujuan yang diinginkan.

Pendekatan *Agentic AI* semakin banyak diterapkan pada arsitektur perangkat lunak modern karena mampu meningkatkan fleksibilitas, otomatisasi, dan kemampuan adaptasi sistem. Dalam konteks penelitian ini, *Agentic AI* tidak dikembangkan sebagai model utama, melainkan dimanfaatkan sebagai komponen dalam sistem untuk mengorkestrasi alur kerja (*workflow*) prediksi kimia. *Agentic AI* berperan sebagai *orchestrator* yang mengatur proses mulai dari validasi struktur molekul, ekstraksi fitur, hingga pemanggilan model bahasa untuk menghasilkan prediksi dan analisis. Dengan demikian, penggunaan *Agentic AI* dalam penelitian ini mendukung integrasi berbagai komponen sistem secara terkoordinasi tanpa mengubah fokus utama penelitian yang berada pada pengembangan sistem berbasis web.

2.2.4 Struktur Polimer dan Properti Material

Polimer merupakan senyawa makromolekul yang tersusun dari unit-unit kecil yang disebut monomer yang terikat melalui ikatan kimia membentuk rantai

panjang. Struktur polimer memiliki karakteristik yang berbeda dibandingkan molekul kecil karena ukuran molekulnya yang besar serta konfigurasi rantai yang kompleks. Struktur ini sangat mempengaruhi sifat fisik, mekanik, dan termal dari material polimer (Jablonka et al., 2023).

Secara umum, struktur polimer dapat direpresentasikan berdasarkan susunan atom dan ikatan kimianya. Dalam kajian komputasi kimia (cheminformatics), struktur molekul sering direpresentasikan dalam bentuk format teks seperti SMILES (Simplified Molecular Input Line Entry System) yang memungkinkan struktur kimia diproses oleh komputer. Representasi ini memudahkan integrasi antara data kimia dengan algoritma machine learning untuk melakukan prediksi berbagai properti material.

Properti material polimer sangat dipengaruhi oleh struktur molekulnya. Faktor seperti panjang rantai polimer, jenis ikatan, percabangan rantai, serta interaksi antar molekul dapat menentukan karakteristik material seperti kekuatan, fleksibilitas, ketahanan panas, dan stabilitas kimia. Oleh karena itu, pemahaman terhadap hubungan antara struktur molekul dan properti material menjadi aspek penting dalam penelitian polymer informatics.

Dalam pendekatan modern, prediksi properti polimer tidak selalu dilakukan melalui eksperimen laboratorium, tetapi dapat dilakukan melalui simulasi komputasi (*in silico*). Pendekatan ini memanfaatkan data struktur molekul sebagai input untuk model Artificial Intelligence guna memperkirakan karakteristik material sebelum dilakukan eksperimen nyata. Metode ini dinilai mampu mengurangi biaya eksperimen serta mempercepat proses penemuan material baru. Dengan demikian, struktur molekul polimer menjadi komponen penting dalam sistem prediksi berbasis AI karena berfungsi sebagai dasar informasi yang akan diproses oleh model machine learning untuk menghasilkan prediksi properti material.

2.2.5 Glass Transition Temperature (Tg)

Glass Transition Temperature (Tg) merupakan salah satu properti termal penting pada material polimer yang menunjukkan suhu transisi ketika material berubah dari kondisi keras dan rapuh (*glassy state*) menjadi lebih lunak dan fleksibel (*rubbery state*). Tidak seperti titik leleh, Tg tidak menunjukkan perubahan fase secara drastis, melainkan perubahan sifat mekanik akibat meningkatnya mobilitas rantai polimer (Schneider et al., 2024).

Nilai Tg memiliki peran yang sangat penting dalam menentukan karakteristik dan aplikasi suatu material polimer. Polimer dengan nilai Tg yang tinggi umumnya memiliki stabilitas termal yang lebih baik dan cocok digunakan pada aplikasi yang membutuhkan ketahanan terhadap suhu tinggi. Sebaliknya, polimer dengan nilai Tg yang rendah cenderung lebih fleksibel dan elastis, sehingga banyak digunakan pada material yang membutuhkan sifat lentur.

Besarnya nilai Tg dipengaruhi oleh berbagai faktor struktural, seperti panjang rantai polimer, jenis ikatan kimia, tingkat percabangan, serta interaksi antar rantai molekul. Semakin kuat interaksi antar molekul dan semakin kaku struktur rantai polimer, maka nilai Tg cenderung meningkat. Oleh karena itu, pemahaman terhadap hubungan antara struktur molekul dan nilai Tg menjadi aspek penting dalam kajian material polimer, khususnya dalam proses perancangan dan pengembangan material baru.

2.2.6 *Simplified Molecular Input Line Entry System* (SMILES)

Simplified Molecular Input Line Entry System (SMILES) merupakan sistem notasi berbasis teks yang digunakan untuk merepresentasikan struktur molekul kimia dalam bentuk untaian karakter ASCII sehingga dapat diproses oleh komputer. Representasi ini menggambarkan atom, ikatan, serta struktur molekul secara linear, sehingga memungkinkan struktur kimia yang kompleks diterjemahkan ke dalam format yang lebih sederhana dan efisien untuk keperluan komputasi.

Dalam perkembangan *cheminformatics* dan *machine learning*, SMILES menjadi salah satu metode representasi molekul yang paling banyak digunakan karena sifatnya yang ringkas, fleksibel, dan kompatibel dengan berbagai algoritma komputasi. Format ini memungkinkan data struktur kimia diintegrasikan langsung ke dalam sistem berbasis kecerdasan buatan untuk keperluan analisis dan prediksi. Penggunaan SMILES dalam penelitian ini berperan sebagai bentuk input utama yang diberikan oleh pengguna ke dalam sistem. Struktur molekul yang dimasukkan dalam format SMILES selanjutnya akan diproses oleh sistem untuk divalidasi, dikonversi menjadi representasi numerik, serta dianalisis untuk menghasilkan prediksi properti material.

2.2.7 RDKit

RDKit merupakan pustaka perangkat lunak *open-source* yang dikembangkan untuk keperluan *cheminformatics* dan pembelajaran mesin molekuler (*molecular machine learning*). Pustaka ini menyediakan berbagai fungsi untuk memproses data kimia, mulai dari pembacaan dan representasi struktur molekul hingga perhitungan berbagai properti fisikokimia (Landrum, 2024). Dalam penelitian ini, RDKit digunakan sebagai komponen utama pada sisi *backend* yang berperan dalam memproses data kimia yang diberikan dalam format SMILES. RDKit bertugas untuk memvalidasi struktur molekul guna memastikan bahwa data yang dimasukkan sesuai dengan aturan kimia yang berlaku. Selain itu, RDKit juga digunakan untuk melakukan normalisasi struktur molekul serta mengekstraksi fitur-fitur penting yang diperlukan dalam proses analisis lebih lanjut.

Penggunaan RDKit memungkinkan konversi data struktur molekul dalam bentuk teks menjadi representasi molekul digital yang dapat diproses secara komputasional. Dengan demikian, RDKit berfungsi sebagai penghubung antara input pengguna dalam bentuk SMILES dengan proses analisis berbasis kecerdasan buatan. Oleh karena itu, RDKit dalam aplikasi web PolyChem digunakan sebagai komponen validasi dan pemrosesan awal yang memastikan kualitas data sebelum diproses lebih lanjut oleh model kecerdasan buatan.

2.2.8 Molecular Fingerprint

Molecular fingerprint merupakan metode representasi numerik dari struktur molekul yang digunakan dalam bidang cheminformatics untuk memudahkan pemrosesan data kimia oleh komputer. Representasi ini mengubah struktur molekul menjadi vektor biner atau numerik yang menggambarkan keberadaan pola tertentu seperti sub-struktur, atom, dan jenis ikatan dalam molekul. Salah satu metode yang umum digunakan adalah Morgan Fingerprint, yang bekerja dengan mengidentifikasi lingkungan atom berdasarkan radius tertentu dan mengubahnya menjadi representasi bit vector.

Penggunaan molecular fingerprint sangat penting dalam sistem berbasis Artificial Intelligence karena memungkinkan struktur kimia yang kompleks diubah menjadi format yang dapat diproses oleh algoritma komputasi. Dengan adanya fingerprint, sistem dapat melakukan analisis kemiripan molekul, klasifikasi, serta prediksi properti material secara lebih efisien. Dalam penelitian ini, molecular fingerprint digunakan sebagai tahap ekstraksi fitur dari struktur molekul berbasis SMILES. Proses ini dilakukan menggunakan library RDKit dengan metode Morgan Fingerprint untuk menghasilkan representasi numerik yang kemudian digunakan dalam analisis lanjutan, seperti perhitungan kemiripan molekul dan pendukung proses reasoning pada sistem *Agentic AI*.

2.2.9 Tanimoto Similarity

Tanimoto Similarity merupakan metode yang digunakan untuk mengukur tingkat kemiripan antara dua buah molekul berdasarkan representasi molecular fingerprint. Metode ini menghitung kesamaan antara dua vektor biner dengan membandingkan jumlah fitur yang sama terhadap total fitur yang dimiliki oleh kedua molekul tersebut. Nilai Tanimoto Similarity berada pada rentang 0 hingga 1, di mana nilai yang mendekati 1 menunjukkan tingkat kemiripan yang tinggi antara dua molekul. Dalam konteks cheminformatics, Tanimoto Similarity banyak digunakan untuk melakukan pencarian senyawa yang memiliki struktur serupa dalam suatu dataset. Pendekatan ini sangat berguna dalam analisis struktur-properti karena molekul yang memiliki kemiripan struktur cenderung memiliki karakteristik fisik dan kimia yang mirip.

Pada penelitian ini, metode Tanimoto Similarity digunakan untuk membandingkan fingerprint molekul input dengan dataset molekul yang tersedia. Hasil perhitungan ini digunakan untuk menampilkan beberapa senyawa yang memiliki kemiripan tertinggi sebagai referensi tambahan bagi pengguna. Dengan demikian, pengguna tidak hanya mendapatkan hasil prediksi, tetapi juga konteks berupa senyawa pembanding yang relevan.

2.2.10 Pipeline Prediksi Properti Polimer Berbasis AI

Dalam sistem prediksi properti material berbasis Artificial Intelligence, diperlukan serangkaian tahapan pemrosesan data yang dikenal sebagai pipeline AI. Pipeline ini menggambarkan bagaimana data mentah diproses secara bertahap hingga menghasilkan informasi yang dapat digunakan oleh pengguna. Dalam konteks *polymer informatics*, pipeline umumnya dimulai dari representasi struktur molekul hingga menghasilkan estimasi properti material.

Pada penelitian ini, pipeline sistem prediksi dimulai dari input struktur molekul dalam format SMILES yang dimasukkan oleh pengguna. Data SMILES kemudian divalidasi menggunakan pustaka RDKit untuk memastikan bahwa struktur molekul dapat diproses secara komputasional. Setelah melalui proses validasi, struktur molekul dikonversi menjadi representasi numerik menggunakan metode *molecular fingerprint* (Morgan Fingerprint) yang berfungsi sebagai fitur kimia untuk analisis lebih lanjut.

Selanjutnya, sistem memanfaatkan pendekatan *Agentic AI* yang mengintegrasikan Large Language Model dengan *chemical tools* untuk melakukan analisis terhadap karakteristik molekul. Pendekatan ini memungkinkan model AI tidak hanya menghasilkan teks, tetapi juga memanfaatkan alat komputasi eksternal untuk meningkatkan akurasi penalaran (*reasoning*). Selain proses prediksi, sistem juga melakukan analisis kemiripan molekul menggunakan metode Tanimoto Similarity. Metode ini digunakan untuk membandingkan fingerprint molekul input dengan data referensi guna menemukan senyawa yang memiliki struktur serupa, sehingga dapat memberikan konteks tambahan terhadap hasil prediksi.

Output dari pipeline ini berupa estimasi properti material, khususnya nilai Glass Transition Temperature (Tg), serta informasi pendukung lainnya yang relevan dengan struktur molekul. Hasil ini disajikan sebagai informasi yang dapat membantu pengguna dalam memahami karakteristik material yang dianalisis. Pipeline ini menjadi dasar dalam perancangan alur aplikasi web PolyChem yang akan diimplementasikan pada tahap pengembangan sistem.

2.2.11 Aplikasi Berbasis Web

Aplikasi berbasis *web* merupakan perangkat lunak yang dapat diakses melalui jaringan internet atau intranet menggunakan perangkat lunak peramban (*web browser*). Pengembangan aplikasi *web* modern umumnya menerapkan arsitektur *Client-Server* yang memisahkan tanggung jawab secara tegas antara sisi antarmuka pengguna (*frontend*) dengan sisi pengelolaan data serta logika bisnis (*backend*).

Integrasi kecerdasan buatan (AI) ke dalam ekosistem aplikasi *web* kini menjadi tren utama dalam pengembangan sistem informasi modern. Sebagaimana dijelaskan oleh (Hou et al., 2024), dalam arsitektur modern, AI tidak lagi sekadar tambahan, melainkan berfungsi sebagai komponen pendukung vital yang meningkatkan kapabilitas dan otomatisasi sistem. Oleh karena itu, pengembangan aplikasi *web* yang terintegrasi dengan layanan AI menuntut perancangan arsitektur sistem yang modular, terstruktur, dan memiliki skalabilitas tinggi agar mudah dikembangkan di masa depan.

2.2.12 *Frontend* dan *Backend*

Frontend (atau sering disebut *client-side*) merupakan lapisan presentasi dari aplikasi *web* yang berinteraksi langsung dengan pengguna, mencakup elemen visual antarmuka dan pengalaman pengguna (*user experience*). Sementara itu, *backend* (atau *server-side*) berfungsi sebagai lapisan logika yang bertanggung jawab atas pengelolaan data, eksekusi algoritma bisnis, serta komunikasi dengan layanan eksternal seperti *database* dan API.

Pemisahan tanggung jawab antara *frontend* dan *backend* (*decoupled architecture*) memungkinkan pengembangan sistem yang lebih terstruktur, mudah

dipelihara (*maintainable*), dan memiliki skalabilitas tinggi (*scalable*) (Richards & Ford, 2020). Dalam penelitian ini, *frontend* dan *backend* dikembangkan secara terpisah namun terintegrasi melalui protokol komunikasi standar untuk mendukung fungsionalitas penuh aplikasi *web* PolyChem.

2.2.13 FastAPI

FastAPI merupakan kerangka kerja (*framework*) berbasis Python yang digunakan untuk membangun layanan *web* dan API dengan performa tinggi. FastAPI dirancang untuk mendukung pengembangan API yang modern, cepat, serta memiliki kemudahan integrasi dengan berbagai sistem eksternal (Safitri & Harkespan, 2024).

Dalam penelitian ini, FastAPI diimplementasikan sebagai fondasi utama di sisi *backend* pada aplikasi *web* PolyChem. Teknologi ini bertugas mengelola logika autentikasi pengguna, orkestrasi komunikasi data, serta menjembatani interaksi antara antarmuka pengguna dengan model AI yang telah tersedia.

2.2.14 React dan Tailwind CSS

React merupakan pustaka (*library*) JavaScript yang dikembangkan oleh Meta untuk membangun antarmuka pengguna berbasis komponen. Teknologi ini memungkinkan pengembangan *frontend* yang dinamis, efisien, dan memiliki pengelolaan *state* yang terstruktur (Meta Open Source, 2024). Sementara itu, Tailwind CSS adalah kerangka kerja (*framework*) CSS yang mengusung pendekatan *utility-first* untuk mempercepat proses penataan gaya (*styling*) antarmuka secara responsif langsung di dalam kelas HTML (Tailwind Labs, 2024).

Kombinasi React dan Tailwind CSS mendukung pengembangan *frontend* yang fleksibel, modern, dan sesuai dengan standar industri aplikasi *web* masa kini. Dalam penelitian ini, kedua teknologi tersebut diimplementasikan secara bersamaan untuk membangun antarmuka pengguna aplikasi PolyChem yang interaktif.

2.2.15 Firebase

Firebase merupakan *platform* pengembangan aplikasi *Backend-as-a-Service* (BaaS) dari Google yang menyediakan berbagai layanan infrastruktur siap pakai, termasuk autentikasi pengguna dan basis data *real-time* (Google Developers, 2024). Salah satu layanan utamanya, *Firebase Authentication*, memungkinkan pengelolaan siklus hidup identitas pengguna (*user lifecycle*) seperti proses *login* dan registrasi secara aman tanpa perlu membangun sistem keamanan dari nol. Dalam penelitian ini, Firebase dimanfaatkan secara spesifik untuk menangani manajemen autentikasi dan penyimpanan data pengguna, dengan batasan implementasi pada akses pengguna standar tanpa hierarki peran (*role*) yang kompleks.

2.2.16 User Interface dan User Experience (UI/UX)

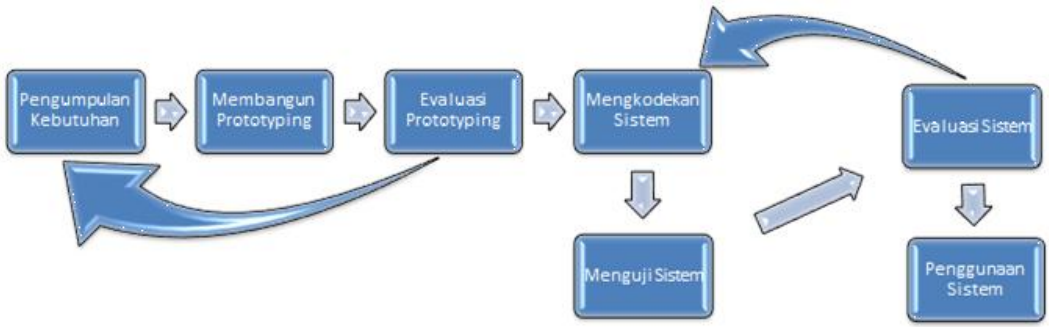
User Interface (UI) dan *User Experience (UX)* merupakan dua aspek fundamental dalam pengembangan aplikasi *web* modern. *UI* berkaitan dengan tata letak visual dan elemen interaksi sistem, sedangkan *UX* berfokus pada keseluruhan pengalaman dan kemudahan yang dirasakan pengguna saat mengoperasikan aplikasi. Desain *UI/UX* yang baik tidak hanya menjamin estetika, tetapi juga meningkatkan efektivitas, efisiensi, dan kepuasan pengguna dalam mengakses fungsionalitas sistem.

Dalam konteks aplikasi saintifik, penggunaan pendekatan desain yang sistematis sangat krusial untuk menyederhanakan visualisasi data yang kompleks. Sebagaimana dikemukakan oleh (Buendía-García & Piris-Ruano, 2025), antarmuka yang intuitif dan responsif berperan signifikan dalam meningkatkan produktivitas peneliti saat berinteraksi dengan alat berbasis AI. Oleh karena itu, perancangan *UI/UX* menjadi salah satu fokus utama dalam penelitian ini guna memastikan aplikasi web PolyChem dapat digunakan dengan mudah oleh pengguna.

2.2.17 Metode *Prototyping*

Tahapan pengembangan sistem dalam penelitian ini mengadopsi model *Prototyping*. Model ini dipilih karena karakteristiknya yang iteratif, memungkinkan pengembang dan pengguna untuk saling berinteraksi dalam menyempurnakan kebutuhan sistem sejak tahap awal, sehingga risiko kesalahan definisi kebutuhan

dapat diminimalkan. Alur kerja metode *Prototyping* yang diacu dalam penelitian ini diilustrasikan pada Gambar 2.1.



Gambar 2. 1 Ilustrasi Metode *Prototyping*
(*Sumber:* https://medium.com/dot-intern/sdlc-metode-prototype)

Menurut (Pressman, 2014), metode *prototyping* merupakan paradigma pengembangan perangkat lunak yang dimulai dengan pengumpulan kebutuhan, diikuti dengan perancangan kilat (*quick design*), pembangunan model awal (*prototype*), evaluasi oleh pengguna, dan penyempurnaan model. Siklus ini berulang hingga sistem dianggap memenuhi spesifikasi yang diinginkan. Metode *prototyping* dipilih dalam penelitian ini karena sangat relevan dengan karakteristik pengembangan aplikasi *web* modern yang dinamis. Pendekatan ini memungkinkan iterasi desain antarmuka (*frontend*) dan logika sistem (*backend*) dilakukan secara berkelanjutan berdasarkan umpan balik (*feedback*) langsung, memastikan bahwa aplikasi web PolyChem yang dibangun benar-benar sesuai dengan kebutuhan pengguna akhir.

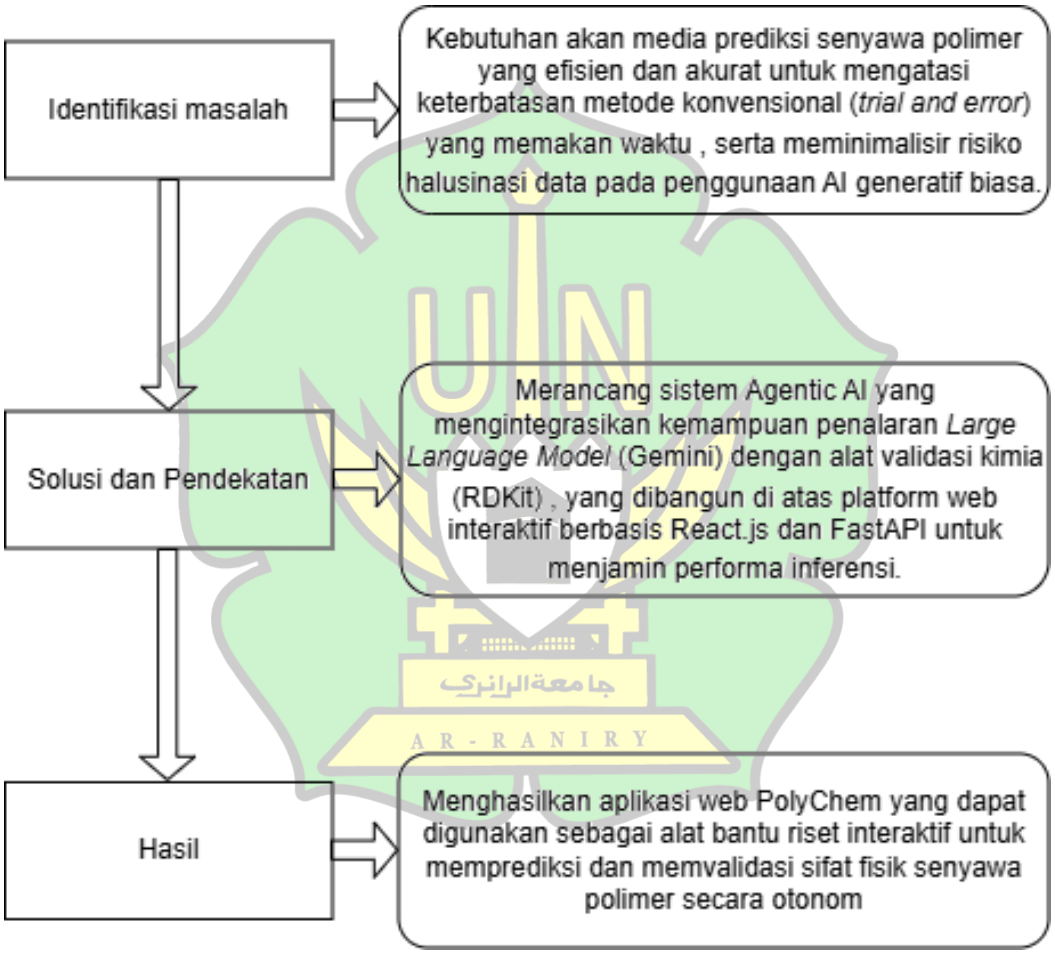
2.2.18 *Black Box Testing*

Black Box Testing adalah metode pengujian perangkat lunak yang berfokus pada validasi fungsionalitas aplikasi tanpa melihat struktur kode internal, detail implementasi, ataupun jalur logika sistem. Pengujian ini dilakukan dengan cara mengamati *input* yang diberikan ke dalam sistem dan memeriksa kesesuaian *output* yang dihasilkan untuk memastikan bahwa setiap fitur berjalan sesuai dengan spesifikasi kebutuhan yang telah ditetapkan (Maulana et al., 2023). Metode ini sangat efektif untuk memverifikasi antarmuka pengguna dan alur logika bisnis dari

sudut pandang pengguna akhir (*end-user*). Selain itu, *Black Box Testing* bertujuan untuk memastikan bahwa sistem bebas dari kesalahan fungsi, kesalahan antarmuka, serta kesalahan pada struktur data atau akses basis data eksternal.

2.3 Kerangka Penelitian

Untuk memberikan gambaran menyeluruh mengenai alur penyelesaian masalah dalam penelitian ini, kerangka penelitian yang menghubungkan urgensi kebutuhan sistem dengan solusi teknis disajikan pada Gambar 2.2.

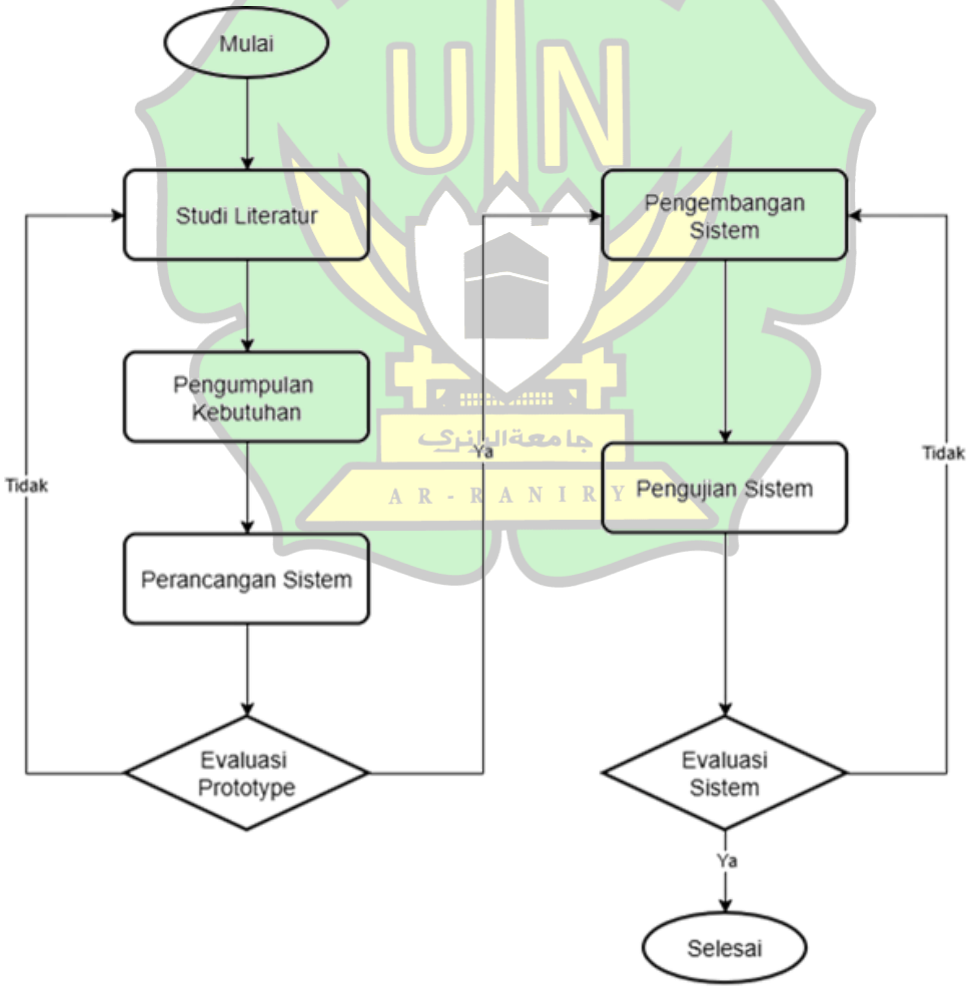


Gambar 2. 2 Kerangka Penelitian

BAB III
METODOLOGI PENELITIAN

3.1 Tahapan Penelitian

Penelitian ini dilaksanakan menggunakan pendekatan *Research and Development* (R&D) yang berfokus pada rekayasa perangkat lunak (*Software Engineering*). Pendekatan ini dipilih untuk menghasilkan produk nyata berupa aplikasi web PolyChem yang terintegrasi dengan kecerdasan buatan. Alur penelitian disusun secara sistematis mulai dari tahap persiapan data hingga penyebaran (*deployment*) sistem ke lingkungan produksi. Secara rinci, tahapan penelitian diilustrasikan pada Gambar 3.1.



Gambar 3. 1 Diagram Alur Penelitian

Penulis mengembangkan aplikasi secara menyeluruh (*full-stack*) yang mencakup sisi klien (*frontend*) dan sisi server (*backend*). Fokus pengembangan aplikasi ini meliputi pembangunan antarmuka interaktif yang berkomunikasi dengan layanan *backend* untuk orkestrasi *Agentic AI*, serta manajemen basis data. Sementara itu, pengembangan atau pelatihan (*training*) model dasar *Large Language Model* (LLM) dan eksperimen laboratorium kimia (*wet lab*) tidak termasuk dalam lingkup pekerjaan penulis, karena sistem ini memanfaatkan layanan model cerdas yang sudah tersedia melalui integrasi API. Rincian dari setiap tahapan penelitian tersebut diuraikan pada subbab-subbab berikut.

3.2 Studi Literatur

Tahap ini merupakan tahap pertama dalam penelitian, sebagaimana ditunjukkan pada Gambar 3.1. Pada tahap ini, penulis mengumpulkan berbagai data dan informasi teknis yang terkait dengan penerapan *Agentic AI*, informatika kimia (cheminformatics), serta arsitektur pengembangan web modern yang relevan dengan penelitian ini. Sumber referensi yang digunakan mencakup jurnal ilmiah nasional maupun internasional, dokumentasi resmi teknologi (documentation), artikel teknis, serta dataset properti polimer yang diperoleh dari repositori publik seperti Kaggle.

Penulis melakukan eksplorasi mendalam terhadap teknologi yang akan diintegrasikan, khususnya dokumentasi pustaka RDKit untuk validasi struktur molekul serta Google Gemini API sebagai mesin kecerdasan buatan utama. Selain itu, penulis juga meninjau penelitian-penelitian terdahulu yang berkaitan dengan metode prediksi polimer secara in silico. Hasil studi literatur menunjukkan bahwa sebagian besar alat prediksi yang tersedia saat ini masih berbasis antarmuka baris perintah (Command Line Interface), sehingga kurang ramah bagi pengguna awam. Selain itu, ditemukan pula bahwa penggunaan Large Language Model (LLM) memiliki potensi menghasilkan halusinasi data apabila tidak didukung oleh mekanisme validasi berbasis algoritma kimia yang terverifikasi.

Melalui studi literatur ini, penulis memperoleh pemahaman mengenai pentingnya membangun arsitektur sistem yang bersifat neuro-symbolic, yaitu menggabungkan kemampuan penalaran AI dengan validasi berbasis aturan kimia yang pasti. Penulis juga mempelajari praktik terbaik (best practice) dalam pengembangan sistem berbasis web menggunakan React.js sebagai *frontend* dan FastAPI sebagai *backend* untuk mendukung efisiensi komputasi. Analisis terhadap celah teknologi (gap analysis) pada sistem prediksi konvensional menjadi dasar dalam perancangan aplikasi web PolyChem. Dengan demikian, studi literatur ini tidak hanya berfungsi sebagai landasan teoretis, tetapi juga sebagai pijakan dalam menentukan pendekatan dan solusi yang akan dikembangkan pada tahap selanjutnya, yaitu pengumpulan kebutuhan sistem.

3.3 Pengumpulan Kebutuhan

Tahap ini merupakan tahap kedua dalam penelitian sesuai dengan alur pada Gambar 3.1, yaitu pengumpulan kebutuhan sistem. Pada tahap ini dilakukan identifikasi terhadap kebutuhan sistem yang akan dikembangkan berdasarkan hasil analisis permasalahan serta studi literatur yang telah dilakukan sebelumnya. Kebutuhan sistem dalam penelitian ini dibagi menjadi dua jenis, yaitu kebutuhan fungsional dan kebutuhan non-fungsional. Kebutuhan fungsional menggambarkan fitur utama yang harus tersedia dalam aplikasi PolyChem, sedangkan kebutuhan non-fungsional berkaitan dengan aspek teknis yang mendukung kinerja sistem.

3.3.1 Kebutuhan Fungsional

Kebutuhan fungsional menggambarkan fungsi dan fitur utama yang akan disediakan oleh aplikasi PolyChem. Fitur-fitur ini dirancang berdasarkan analisis masalah kesulitan peneliti dalam memprediksi properti polimer secara manual. Rincian fitur dituliskan dalam bentuk *user story* dengan format "Sebagai <aktor>, saya ingin <aktivitas> sehingga <tujuan/manfaat>". Pada Tabel 3.1 dijelaskan fitur-fitur yang akan dikembangkan dalam aplikasi web PolyChem.

Tabel 3. 1 Fitur Aplikasi

No	Fitur	Deskripsi
1	Autentikasi	Fitur ini menangani keamanan akses aplikasi melalui mekanisme *login* dan registrasi. Fitur ini memastikan hanya pengguna terdaftar yang dapat mengakses riwayat prediksi dan menyimpan koleksi data pribadi.
2	Prediksi Polimer (*Main*)	Fitur utama yang memungkinkan pengguna memasukkan struktur kimia dalam format SMILES. Sistem akan memvalidasi struktur tersebut menggunakan RDKit dan mengirimkannya ke layanan *Agentic AI* untuk mendapatkan prediksi properti polimer (seperti *Glass Transition Temperature*).
3	Riwayat (*History*)	Fitur ini berfungsi menyimpan seluruh log aktivitas prediksi yang pernah dilakukan oleh pengguna. Pengguna dapat melihat kembali hasil analisis sebelumnya tanpa perlu melakukan input ulang.
4	Koleksi (*Library*)	Fitur ini memungkinkan pengguna untuk menyimpan atau "menandai" senyawa polimer tertentu ke dalam koleksi pribadi agar mudah diakses di kemudian hari.
5	Dasbor Utama	Fitur ini berfungsi sebagai tampilan antarmuka awal yang menyajikan ringkasan aktivitas pengguna dan navigasi cepat menuju formulir prediksi atau daftar riwayat.

Berikut adalah rincian *User Stories* dari kebutuhan fitur di atas yang dapat dilihat pada Tabel 3.2.

Tabel 3. 2 *User Story*

No	**Fitur**	**Judul *User Story***	***User Story***
1	Autentikasi	Login & Register	Sebagai pengguna, saya ingin mendaftar dan masuk ke dalam sistem, sehingga saya dapat menyimpan dan mengakses data prediksi saya secara aman.
2	Prediksi	Input SMILES	Sebagai pengguna, saya ingin memasukkan struktur SMILES polimer, sehingga sistem dapat memvalidasi dan memproses struktur kimia tersebut secara otomatis.
3	Prediksi	Lihat Hasil AI	Sebagai pengguna, saya ingin melihat hasil prediksi properti polimer, sehingga saya dapat memahami karakteristik material tanpa melakukan uji laboratorium secara langsung.
4	Riwayat	Akses Riwayat	Sebagai pengguna, saya ingin melihat riwayat prediksi yang pernah dilakukan, sehingga saya dapat mengakses kembali hasil analisis tanpa perlu melakukan input ulang.
5	Koleksi	Simpan ke Library	Sebagai pengguna, saya ingin menyimpan senyawa polimer ke dalam koleksi pribadi, sehingga saya dapat

			mengaksesnya kembali dengan mudah di kemudian hari.

3.3.2 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional menjelaskan batasan teknis terkait perangkat lunak dan perangkat keras yang digunakan dalam proses pengembangan aplikasi. Dalam penelitian ini, perangkat lunak yang digunakan untuk membangun aplikasi web PolyChem dapat dilihat pada Tabel 3.3.

Tabel 3. 3 Perangkat Lunak

No	Nama	Versi / Keterangan	Deskripsi
1	Visual Studio Code	1.133.0	Digunakan sebagai *code editor*
2	Google Chrome	151.0.7922.109	Digunakan untuk menjalankan aplikasi web
3	Figma	126.7.10	Digunakan untuk desain *UI/UX*
4	Python	3.10.11	Bahasa pemrograman *backend*
5	Node.js	v23.8.0	Runtime *frontend*
6	React.js	19.2.0	Library *frontend*
7	FastAPI	0.124.2	Framework *backend*
8	Firebase	12.6.0	*Database* dan *autentikasi*
9	Postman	12.22.2	Pengujian *API*
10	Git & Github	-	*Version Control System*

Perangkat keras yang digunakan penulis dalam pengembangan aplikasi web PolyChem pada penelitian ini dapat dilihat pada Tabel 3.4.

Tabel 3. 4 Perangkat Keras

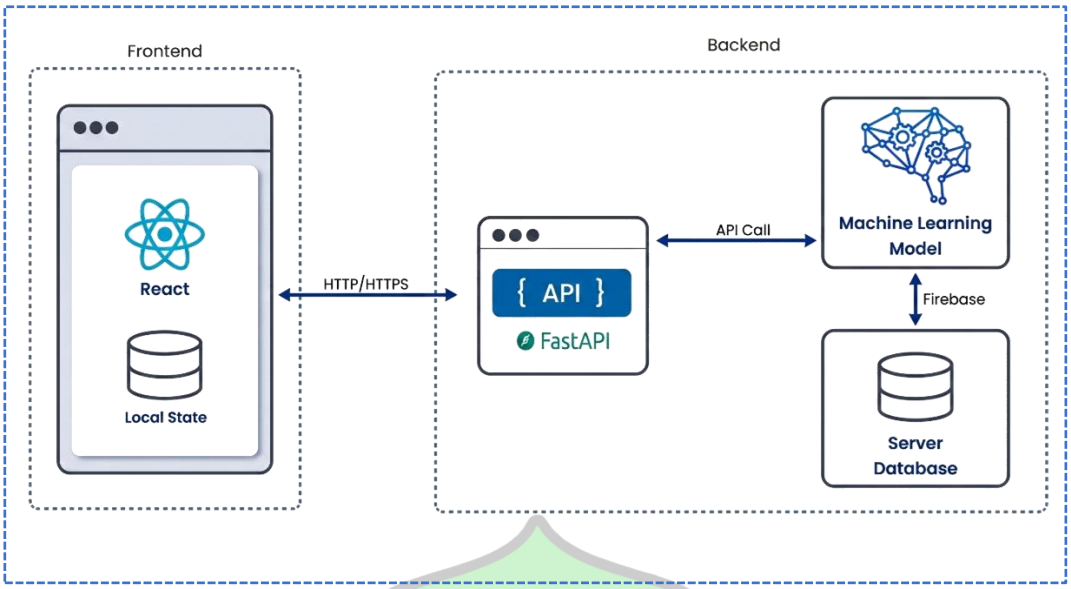
No	Nama	Spesifikasi
1	Laptop / PC	ASUS Vivobook Pro 14 OLED
2	Sistem Operasi	Windows 11
3	Processor (CPU)	Intel Core i5
4	RAM	8 GB
5	Penyimpanan	SSD 512 GB

3.4 Perancangan Sistem

Tahap ini merupakan tahap ketiga dalam penelitian sesuai dengan alur pada Gambar 3.1, yaitu perancangan sistem. Pada tahap ini dilakukan proses perancangan terhadap aplikasi web PolyChem berdasarkan kebutuhan yang telah diidentifikasi pada tahap sebelumnya. Perancangan sistem dilakukan menggunakan pendekatan prototyping, di mana rancangan sistem dibuat dalam bentuk model awal (prototype) yang dapat dievaluasi sebelum masuk ke tahap pengembangan. Proses ini bertujuan untuk memastikan bahwa sistem yang dirancang telah sesuai dengan kebutuhan pengguna. Perancangan sistem dalam penelitian ini mencakup beberapa aspek, yaitu perancangan kebutuhan pengguna (*user requirement*), pemodelan sistem menggunakan UML, perancangan arsitektur sistem, serta desain antarmuka pengguna (*user interface*).

3.4.1 Arsitektur Sistem

Aplikasi web PolyChem dirancang menggunakan arsitektur *Client-Server* yang terpisah (*decoupled architecture*) untuk menjamin fleksibilitas dan skalabilitas pengembangan. Sisi klien (*frontend*) dibangun menggunakan React.js yang bertugas menangani interaksi pengguna, sedangkan sisi server (*backend*) dibangun menggunakan FastAPI yang berfungsi mengorkestrasi logika bisnis dan layanan AI. Ilustrasi arsitektur sistem secara keseluruhan ditampilkan pada Gambar 3.2.



Gambar 3. 2 Arsitektur aplikasi web PolyChem

a) *Frontend* (Sisi Klien): Komponen ini dibangun menggunakan React.js. Di dalam diagram, komponen ini berinteraksi dengan pengguna dan mengelola penyimpanan lokal (*local state*) untuk efisiensi antarmuka. Komunikasi ke luar dilakukan melalui protokol HTTP/HTTPS untuk mengirimkan permintaan data ke server.

b) *Backend* (Sisi Server): Blok besar di sisi kanan merepresentasikan lingkungan server (*server-side environment*) yang menaungi tiga komponen krusial yang saling terintegrasi:

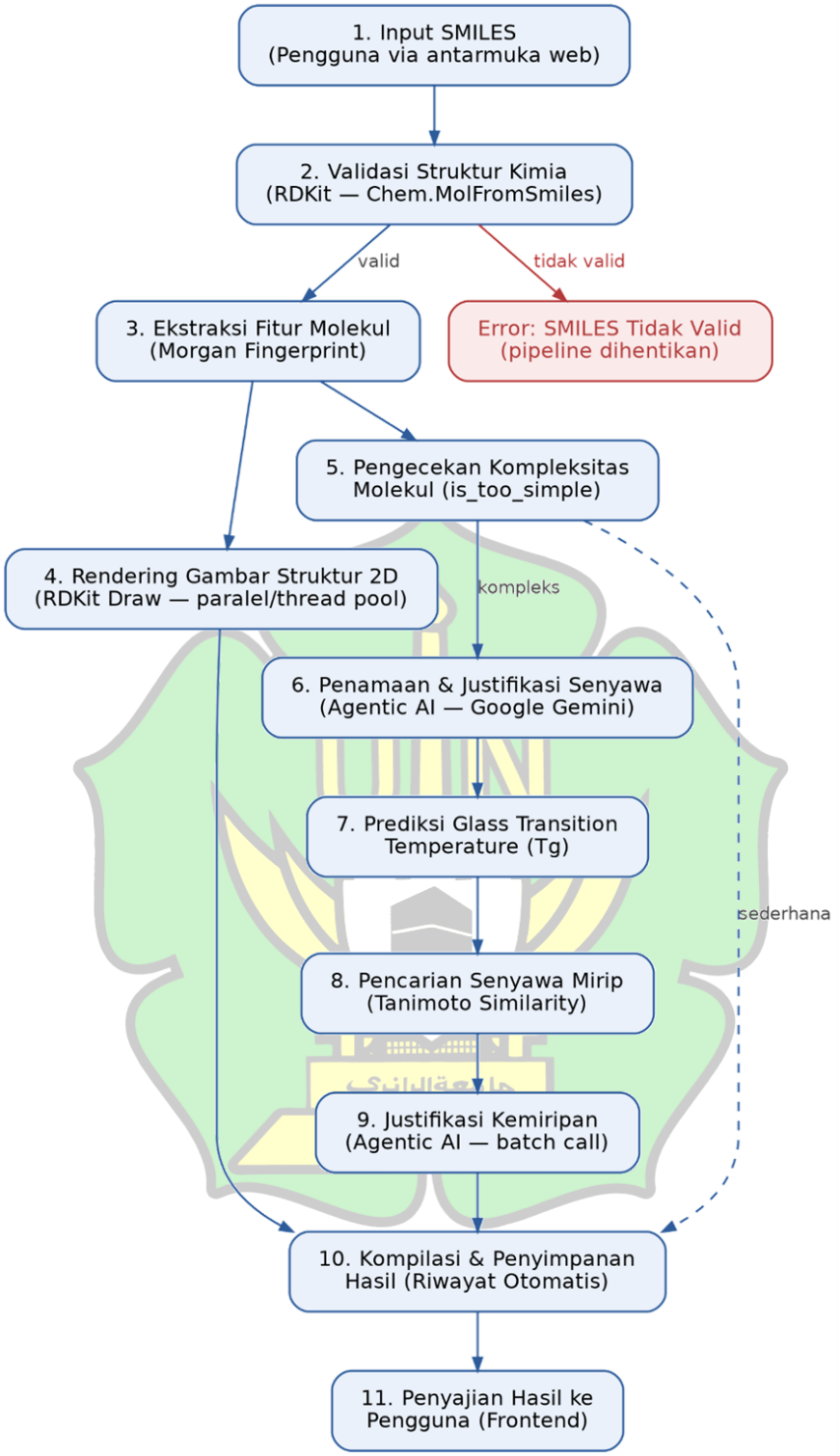
1. FastAPI Wrapper: Bertindak sebagai gerbang utama (*API Gateway*) yang menerima permintaan dari *frontend*.
2. Machine Learning Model: Merupakan otak dari sistem (*Agentic AI*) yang memproses logika prediksi kimia. Komponen ini berkomunikasi dengan FastAPI melalui mekanisme *internal API Call* atau pemanggilan fungsi langsung.
3. Server *Database*: Menggunakan layanan Firebase (Firestore) yang terhubung langsung dalam ekosistem *backend*. Ini berfungsi sebagai tempat penyimpanan permanen untuk data pengguna, riwayat prediksi, dan pustaka senyawa.

Pendekatan arsitektur terpusat di *backend* ini memastikan bahwa seluruh logika bisnis dan koneksi *database* terlindungi di sisi server, sementara *frontend* hanya fokus pada presentasi data visual (Richards & Ford, 2020).

3.4.2 Perancangan Alur Pipeline Prediksi

Selain arsitektur *client-server* yang telah dijelaskan sebelumnya, komponen inti dari aplikasi web PolyChem terletak pada rangkaian proses di sisi backend yang mengubah masukan berupa notasi SMILES menjadi keluaran berupa prediksi properti termal polimer beserta justifikasi ilmiahnya. Rangkaian proses ini dirancang sebagai sebuah pipeline yang menggabungkan dua pendekatan komputasi yang berbeda sifat, yaitu komputasi kimia deterministik menggunakan RDKit dan penalaran generatif menggunakan model bahasa besar (*Large Language Model/LLM)* Google Gemini sebagai komponen *Agentic AI*. Pendekatan gabungan ini dikenal sebagai pendekatan *neuro-symbolic*, di mana komponen simbolik (RDKit) berperan memastikan validitas dan akurasi kimiawi, sementara komponen neural (Gemini) berperan menghasilkan narasi ilmiah dan estimasi properti yang tidak dapat dihitung secara analitis murni.

Alur pipeline prediksi pada aplikasi web PolyChem dirancang melalui sebelas tahapan berurutan, sebagaimana digambarkan pada Gambar 3.3.



Gambar 3. 3 Diagram Pipeline prediksi

Penjelasan tiap tahap pada pipeline tersebut adalah sebagai berikut:

1. Input SMILES. Pengguna memasukkan notasi SMILES yang merepresentasikan struktur molekul senyawa melalui antarmuka *frontend*

aplikasi web PolyChem. Data ini dikirimkan ke *backend* dalam format JSON melalui protokol HTTP.

2. Validasi Struktur Kimia. *Backend* melakukan validasi terhadap notasi SMILES yang diterima menggunakan pustaka RDKit, khususnya fungsi *parser* dan *sanitizer* bawaan (MolFromSmiles). Tahap ini berfungsi sebagai gerbang (*gate*) utama pipeline: apabila struktur tidak valid secara sintaksis maupun kimiawi (misalnya kesalahan valensi atau notasi cincin yang tidak tertutup), proses dihentikan dan sistem mengembalikan pesan galat kepada pengguna sebelum memasuki tahap komputasi berikutnya. Perancangan ini bertujuan mencegah pemrosesan lebih lanjut termasuk pemanggilan model AI terhadap masukan yang secara kimiawi tidak mungkin ada, sehingga menghemat sumber daya komputasi dan mencegah potensi keluaran yang menyesatkan (*hallucination*) dari model AI akibat data masukan yang tidak valid.
3. Ekstraksi Fitur Molekul. Struktur molekul yang telah tervalidasi dikonversi menjadi representasi numerik menggunakan metode *Morgan Fingerprint*, yang merepresentasikan lingkungan atom dalam bentuk vektor biner. Representasi ini menjadi dasar bagi tahap pencarian kemiripan struktur pada tahap selanjutnya.
4. Rendering Gambar Struktur 2D. Secara paralel dengan tahap berikutnya, sistem menghasilkan visualisasi dua dimensi dari struktur molekul menggunakan modul *drawing* RDKit. Proses ini bersifat deterministik sepenuhnya tidak melibatkan model AI generatif sehingga struktur yang sama akan selalu menghasilkan visualisasi yang identik.
5. Pengecekan Kompleksitas Molekul. Sistem memeriksa kompleksitas struktur berdasarkan jumlah atom penyusun. Molekul yang tergolong sangat sederhana (misalnya senyawa dengan atom kurang dari ambang batas tertentu) tidak diteruskan ke tahap penalaran AI, melainkan diberi label dan justifikasi generik, karena penalaran mendalam dinilai tidak diperlukan untuk struktur yang trivial.
6. Penamaan dan Justifikasi Senyawa. Untuk molekul dengan kompleksitas memadai, sistem memanggil model Agentic AI utama (Google Gemini) untuk menghasilkan nama senyawa serta justifikasi ilmiah mengenai keunikan strukturnya, mencakup aspek gugus fungsi, fleksibilitas rantai, dan kekakuan

struktural. Apabila terjadi gangguan jaringan (*timeout*) atau limitasi kuota pada agen utama, sistem akan menerapkan mekanisme *fail-fast* dan mengalihkan tugas komputasi secara otonom ke penyedia AI cadangan (Groq) sebelum memicu *fallback* heuristik lokal berbasis RDKit.

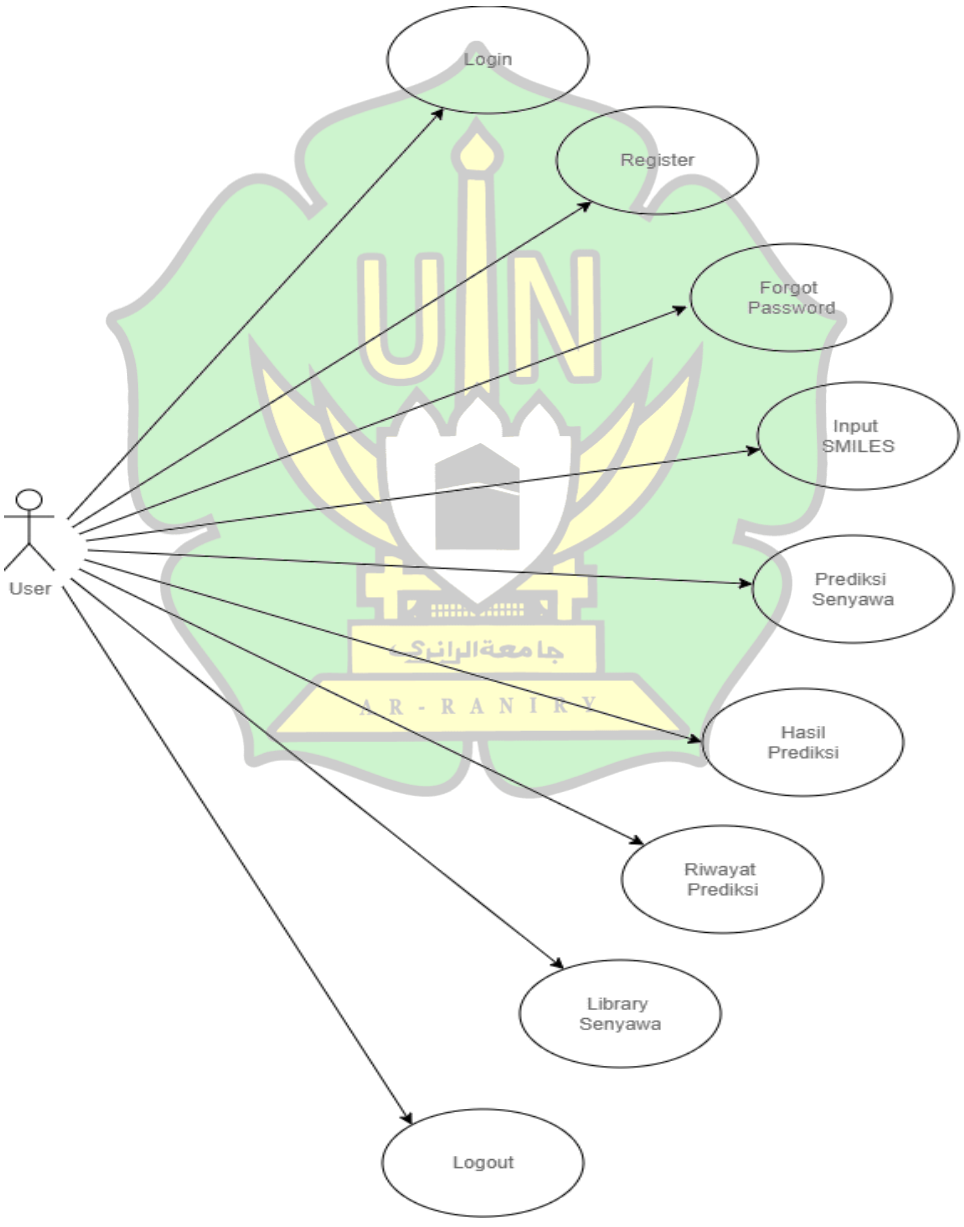
7. Prediksi *Glass Transition Temperature* (Tg). Nilai Tg diprediksi melalui salah satu dari tiga jalur, tergantung ketersediaan data: (a) apabila SMILES ditemukan persis pada dataset referensi, nilai Tg diambil langsung dari data eksperimen; (b) apabila tidak ditemukan, sistem memanggil model Gemini untuk memprediksi nilai Tg berdasarkan fitur struktural; (c) sebagai mekanisme cadangan (*fallback*) apabila pemanggilan model AI gagal atau melampaui batas waktu, sistem menghitung estimasi Tg menggunakan heuristik berbasis deskriptor RDKit (jumlah cincin, ikatan yang dapat berotasi, dan massa molekul).
8. Pencarian Senyawa Mirip. Sistem membandingkan fingerprint molekul masukan dengan seluruh fingerprint pada dataset referensi menggunakan metode *Tanimoto Similarity*, kemudian mengambil tiga senyawa dengan tingkat kemiripan tertinggi.
9. Justifikasi Kemiripan. Untuk ketiga senyawa dengan kemiripan tertinggi tersebut, sistem memanggil model *Agentic AI* untuk menghasilkan justifikasi naratif mengenai motif struktural yang menjadi dasar kemiripan dengan senyawa masukan.
10. Kompilasi dan Penyimpanan Hasil. Seluruh keluaran dari tahap-tahap sebelumnya (data senyawa baru dan senyawa-senyawa serupa) dikompilasi menjadi satu struktur data, kemudian dicatat secara otomatis ke dalam riwayat pencarian pengguna.
11. Penyajian Hasil. Struktur data hasil kompilasi dikembalikan ke *frontend* dalam format JSON untuk dirender menjadi antarmuka hasil prediksi yang informatif bagi pengguna.

Perancangan pipeline ini menekankan prinsip keandalan berlapis (*layered reliability*): setiap tahap yang melibatkan pemanggilan model AI generatif (tahap 6, 7, dan 9) dirancang memiliki mekanisme cadangan berbasis komputasi

deterministik RDKit, sehingga kegagalan atau keterbatasan pada layanan AI eksternal tidak menyebabkan kegagalan total pada aplikasi web PolyChem.

3.4.3 Pemodelan Fungsional (*Use Case Diagram*)

Untuk memetakan interaksi antara pengguna dengan fitur-fitur aplikasi web PolyChem, digunakan pemodelan visual berupa *Use Case Diagram*. Diagram ini menggambarkan hak akses peneliti dalam menggunakan fitur prediksi dan manajemen data. Rancangan *Use Case Diagram* dapat dilihat pada Gambar 3.4 berikut.



Gambar 3. 4 *Use Case* Diagram PolyChem

Sistem dirancang dengan satu aktor utama, yaitu Peneliti, yang memiliki akses ke lima fungsi utama sistem sebagai berikut:

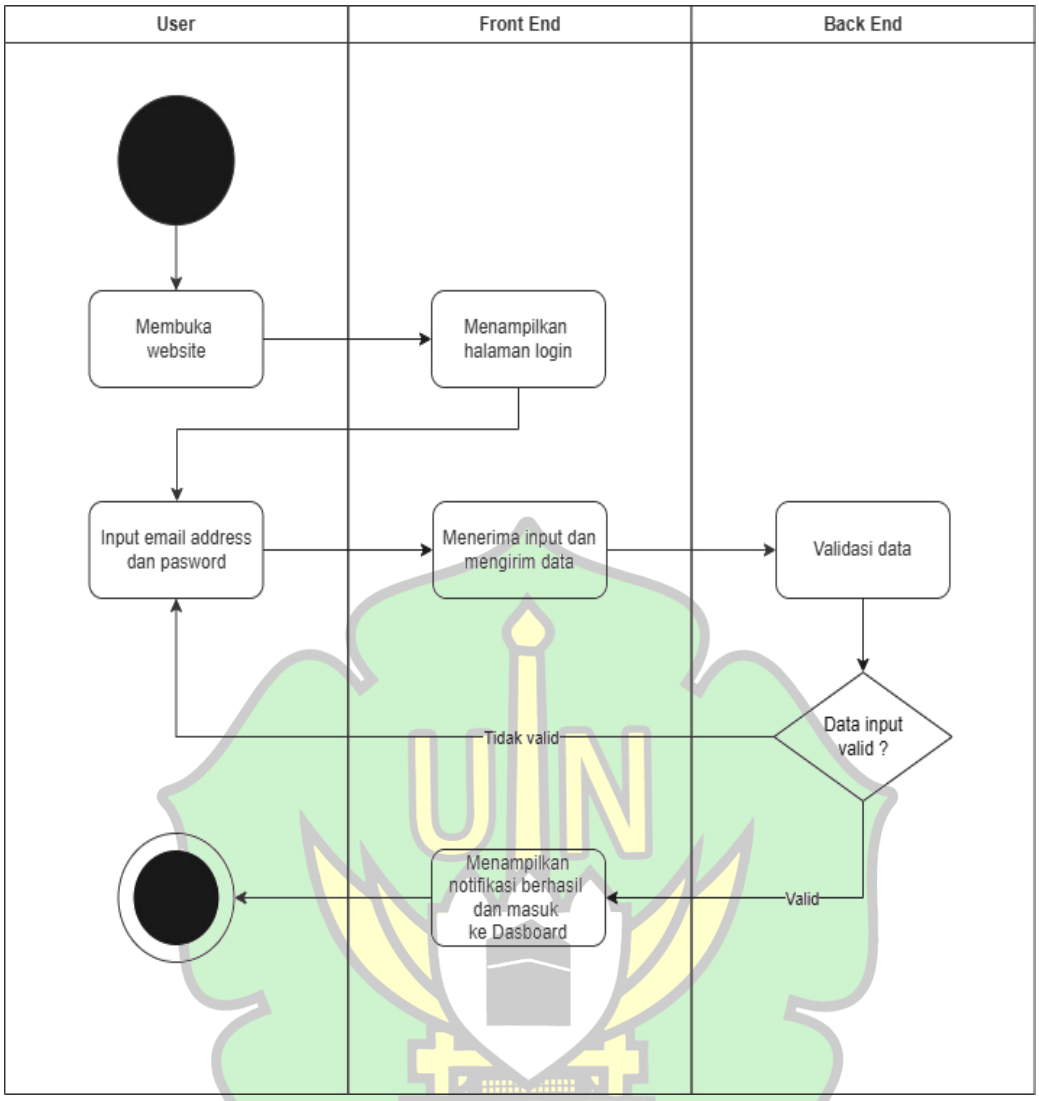
1. Login/Register: Aktor melakukan autentikasi untuk mendapatkan hak akses masuk ke dalam sistem.
2. Input SMILES: Aktor memasukkan format teks kimia untuk diproses oleh sistem.
3. Lihat Hasil Prediksi: Aktor menerima output properti polimer yang dihasilkan oleh *Agentic AI*.
4. Kelola Library: Aktor dapat menyimpan atau menghapus data senyawa dari koleksi pribadi.
5. Lihat History: Aktor dapat memantau riwayat aktivitas prediksi yang pernah dilakukan.

3.4.4 Perancangan Alur Aktivitas (*Activity Diagram*)

Activity Diagram dirancang untuk menggambarkan alur kerja (*workflow*) prosedural dari fitur-fitur utama aplikasi web PolyChem. Diagram ini menjelaskan urutan aktivitas yang dilakukan oleh peneliti dan respons yang diberikan oleh sistem.

a. Activity Login

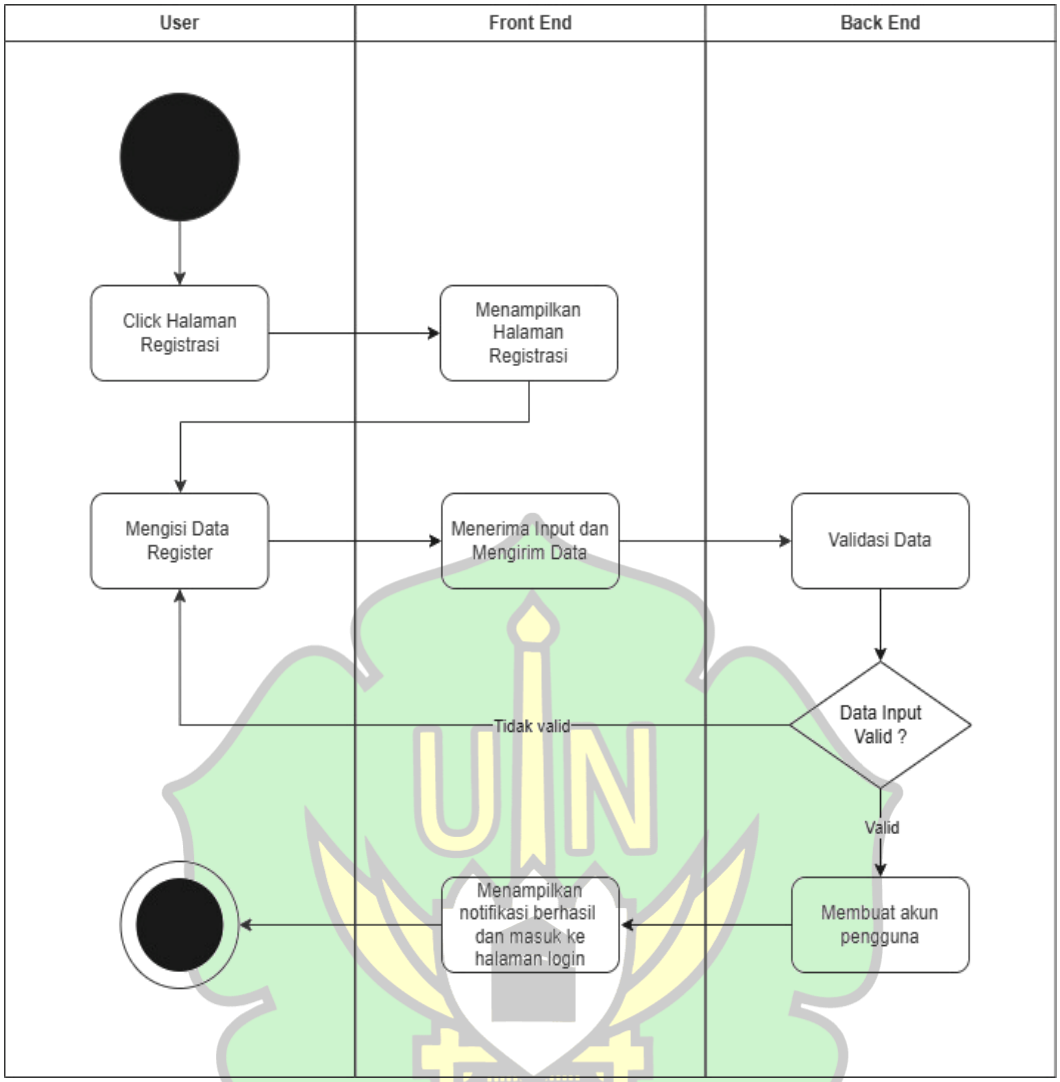
Activity Login menggambarkan proses pengguna dalam melakukan login ke sistem. Pengguna memasukkan email dan password, kemudian sistem melakukan validasi data. Jika data valid, pengguna dapat mengakses halaman utama sistem, sedangkan jika data tidak valid maka sistem akan menampilkan pesan kesalahan.



Gambar 3. 5 Diagram Login – *Activity*

b. Activity Registrasi

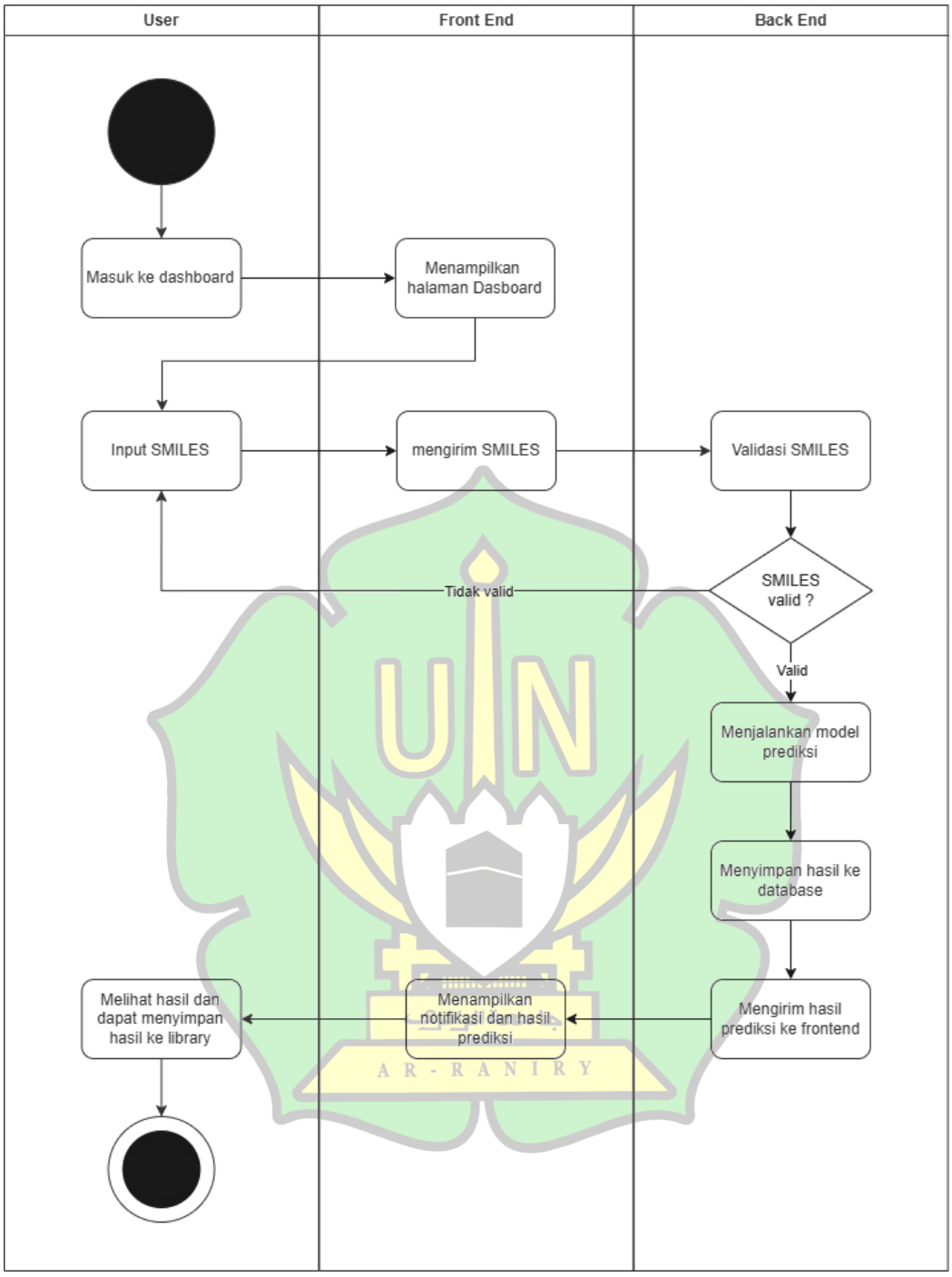
Activity Registrasi menggambarkan proses pendaftaran pengguna baru. Pengguna mengisi data registrasi, kemudian sistem memvalidasi data tersebut. Jika data valid, akun pengguna akan dibuat dan pengguna diarahkan ke halaman login.



Gambar 3. 6 Diagram Registrasi – *Activity*

c. Activity Prediksi Senyawa

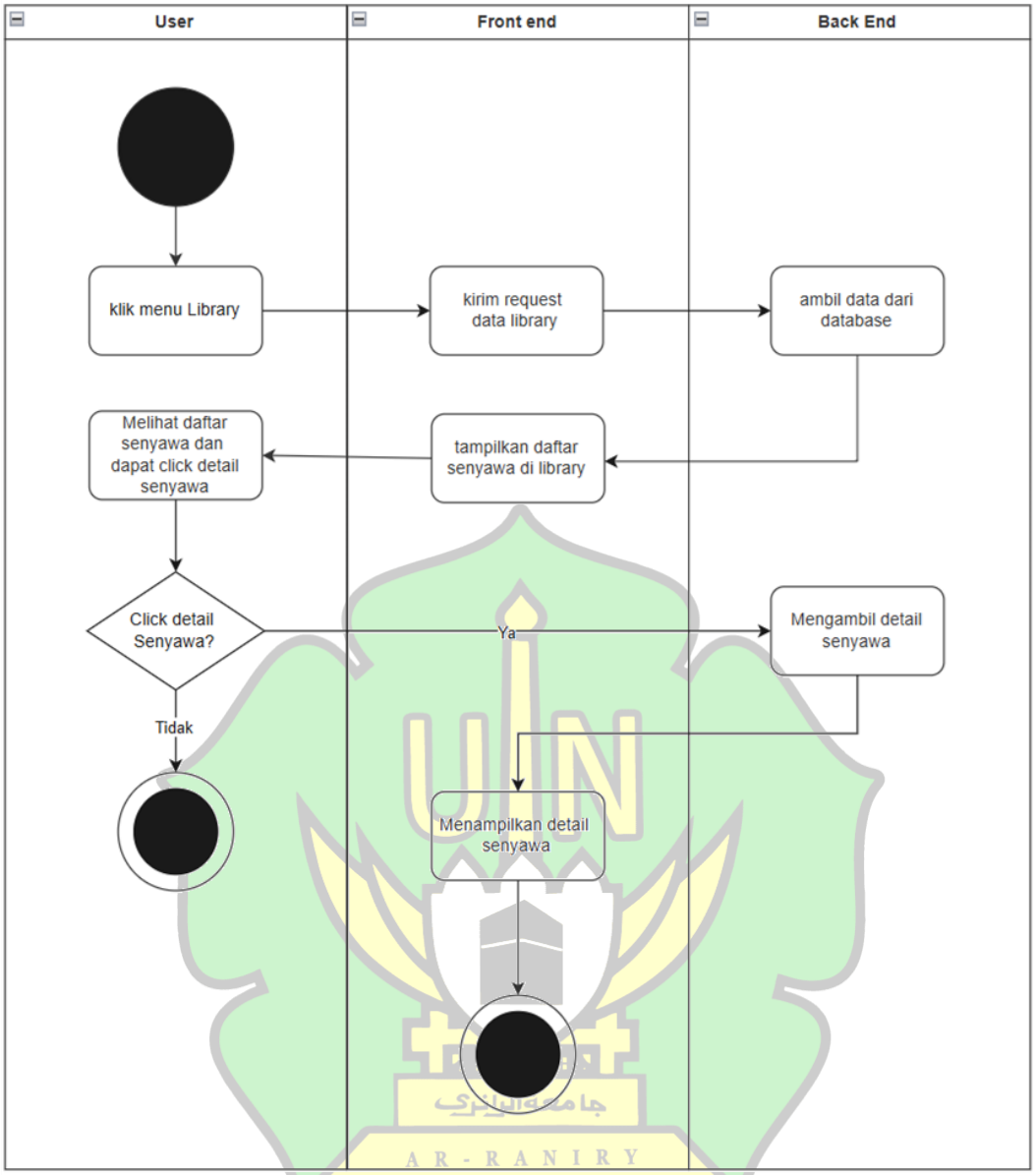
Activity Prediksi Senyawa menggambarkan proses utama sistem, yaitu prediksi senyawa berdasarkan input SMILES. Pengguna memasukkan data SMILES, kemudian sistem memproses data tersebut dan menampilkan hasil prediksi senyawa kepada pengguna. Hasil prediksi juga dapat disimpan ke dalam riwayat atau library.



Gambar 3. 7 Diagram Prediksi Senyawa – *Activity*

d. Activity Library

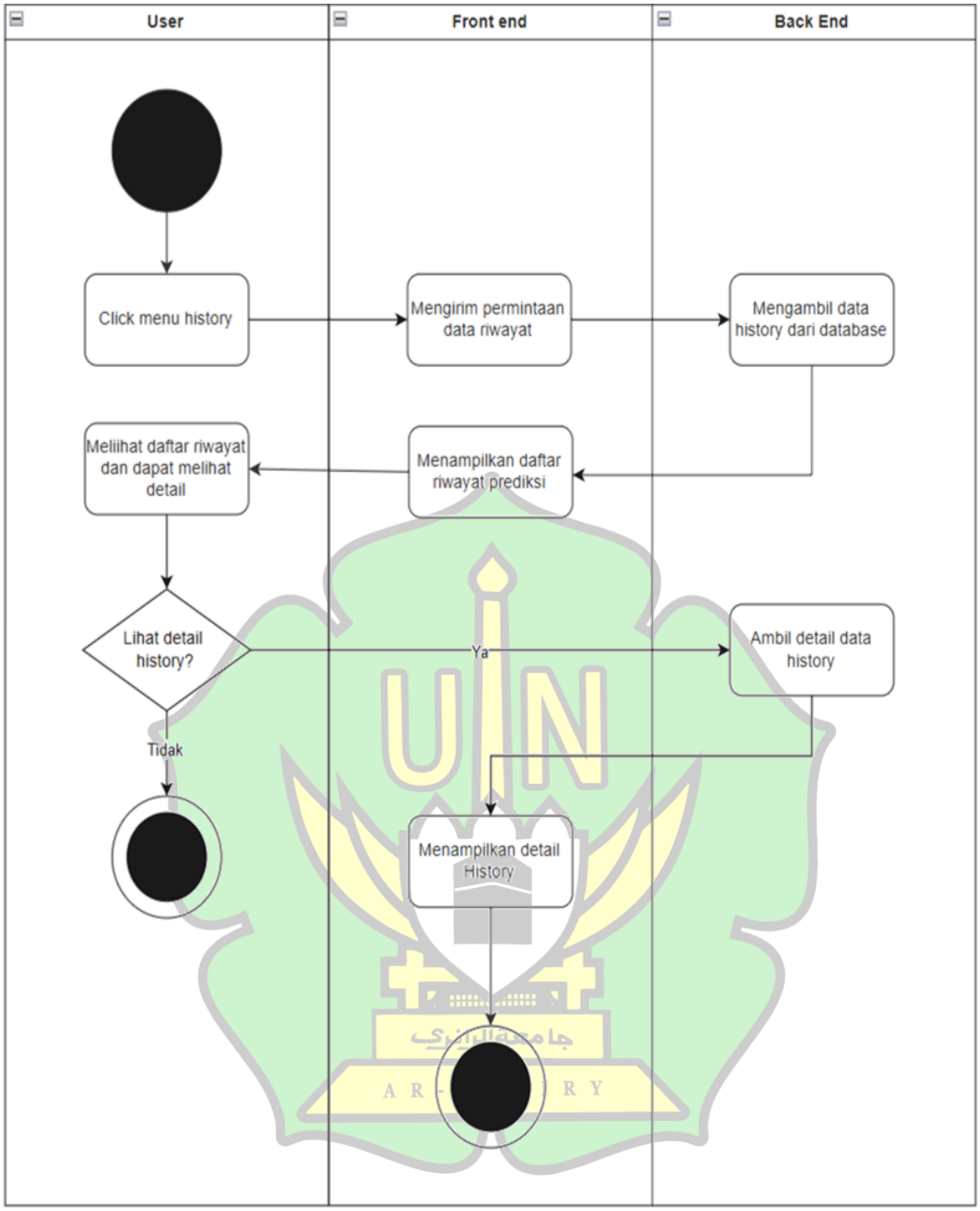
Activity Library menggambarkan proses pengguna dalam melihat data senyawa yang tersimpan. Pengguna membuka menu library, kemudian sistem mengambil data dari *database* dan menampilkannya kepada pengguna. Pengguna juga dapat melihat detail senyawa yang dipilih.



Gambar 3. 8 Diagram Library – *Activity*

e. Activity History

Activity History menggambarkan proses pengguna dalam melihat riwayat prediksi senyawa. Sistem mengambil data riwayat dari *database* dan menampilkannya kepada pengguna. Pengguna dapat melihat detail hasil prediksi yang pernah dilakukan sebelumnya.



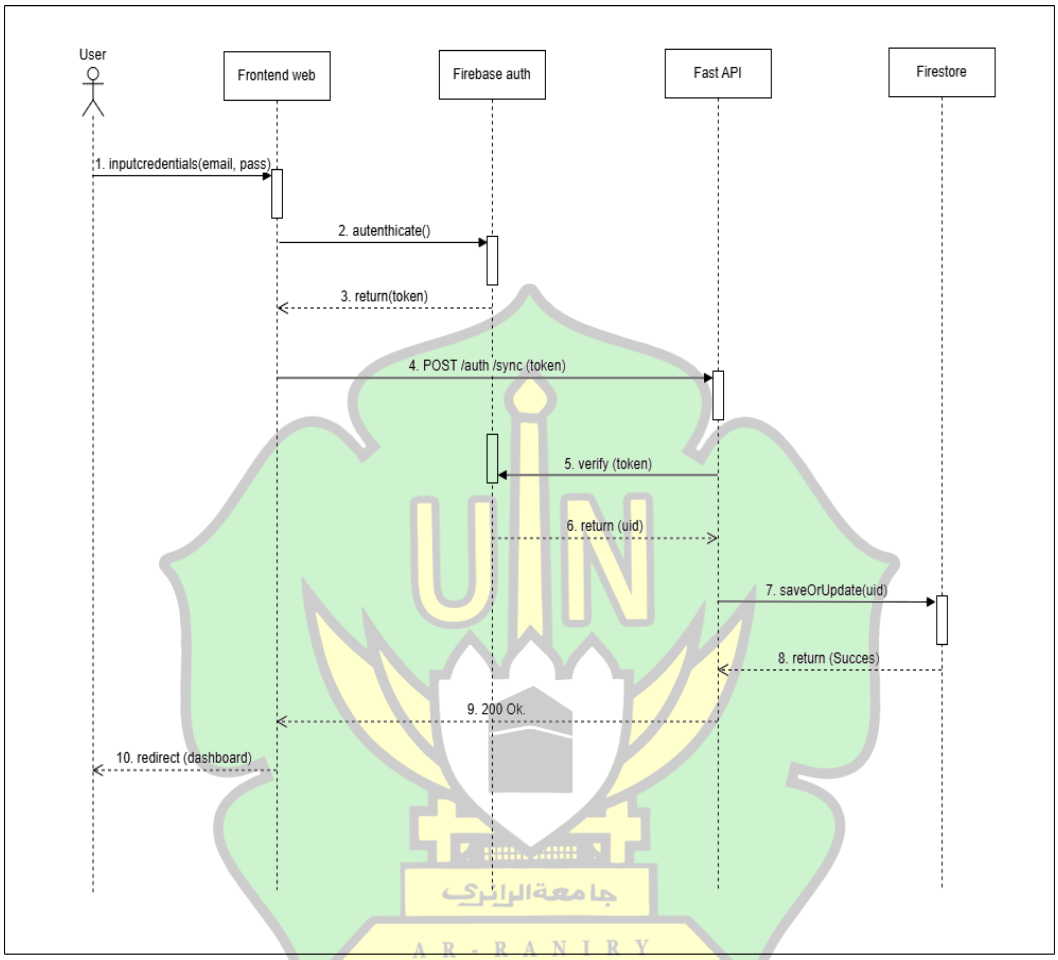
Gambar 3. 9 Diagram History – *Activity*

3.4.5 Perancangan Interaksi Objek (*Sequence Diagram*)

Sequence Diagram memvisualisasikan interaksi antar objek di dalam sistem berdasarkan urutan waktu. Diagram ini memperjelas bagaimana data mengalir dari antarmuka (*Frontend*) menuju *Backend* dan layanan eksternal (*Third-party Services*).

a. Sequence Diagram: Autentikasi (Login & Register)

Diagram ini menjelaskan alur masuk dan pendaftaran pengguna menggunakan layanan *Firebase Authentication* yang disinkronkan dengan *Backend API* dan *Firestore* .

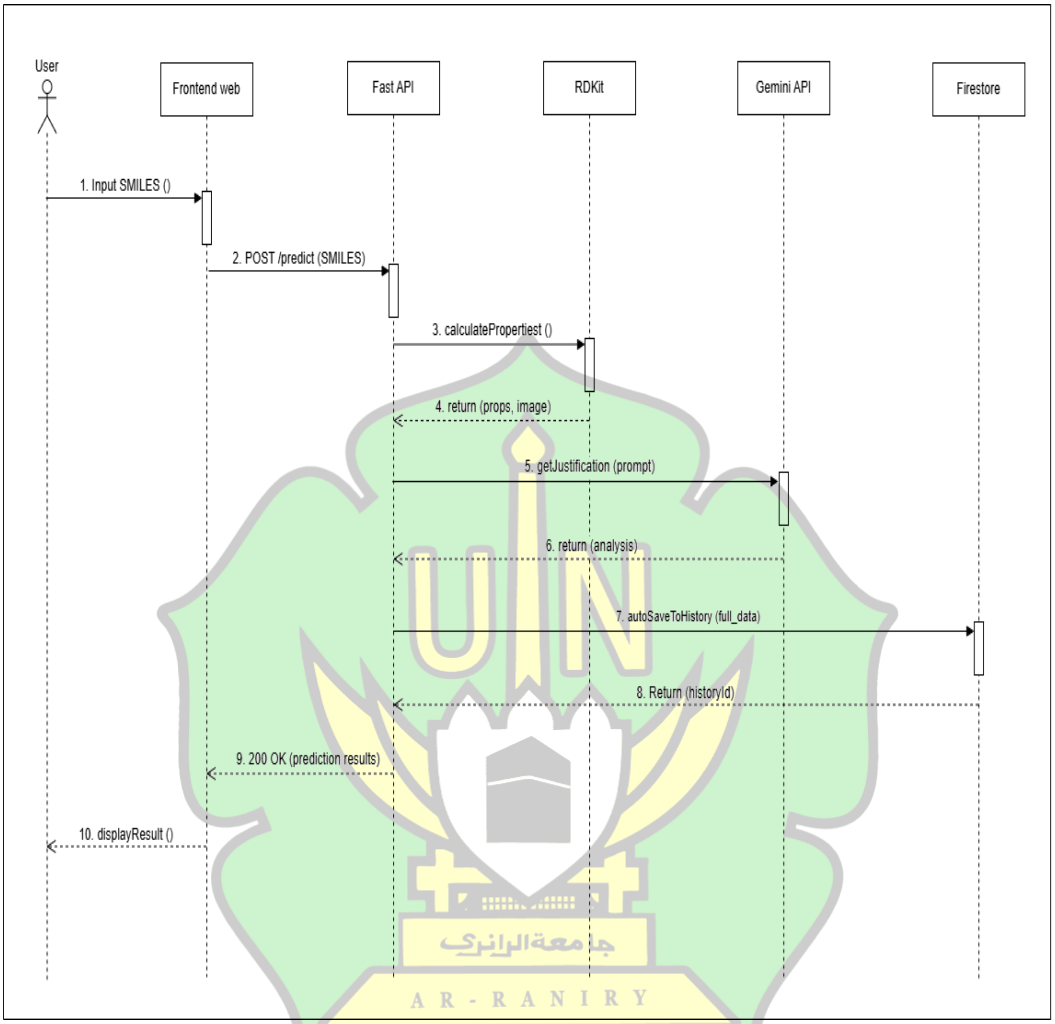


Gambar 3. 10 *Sequence Diagram* – Autentikasi

Gambar di atas menunjukkan alur autentikasi pengguna yang mengintegrasikan layanan eksternal Firebase Authentication. Proses dimulai ketika pengguna memasukkan kredensial melalui *Frontend* Web, yang kemudian diteruskan ke Firebase untuk validasi. Setelah mendapatkan token identitas, *Frontend* mengirimkannya ke FastAPI untuk diverifikasi kembali guna memastikan keamanan sesi sebelum akhirnya sistem melakukan sinkronisasi atau pembaruan data profil pengguna ke dalam koleksi *users* di Firestore.

b. Sequence Diagram: Prediksi Senyawa & Auto-History

Diagram ini merupakan representasi proses inti sistem yang melibatkan orkestrasi antara algoritma komputasi dan kecerdasan buatan .

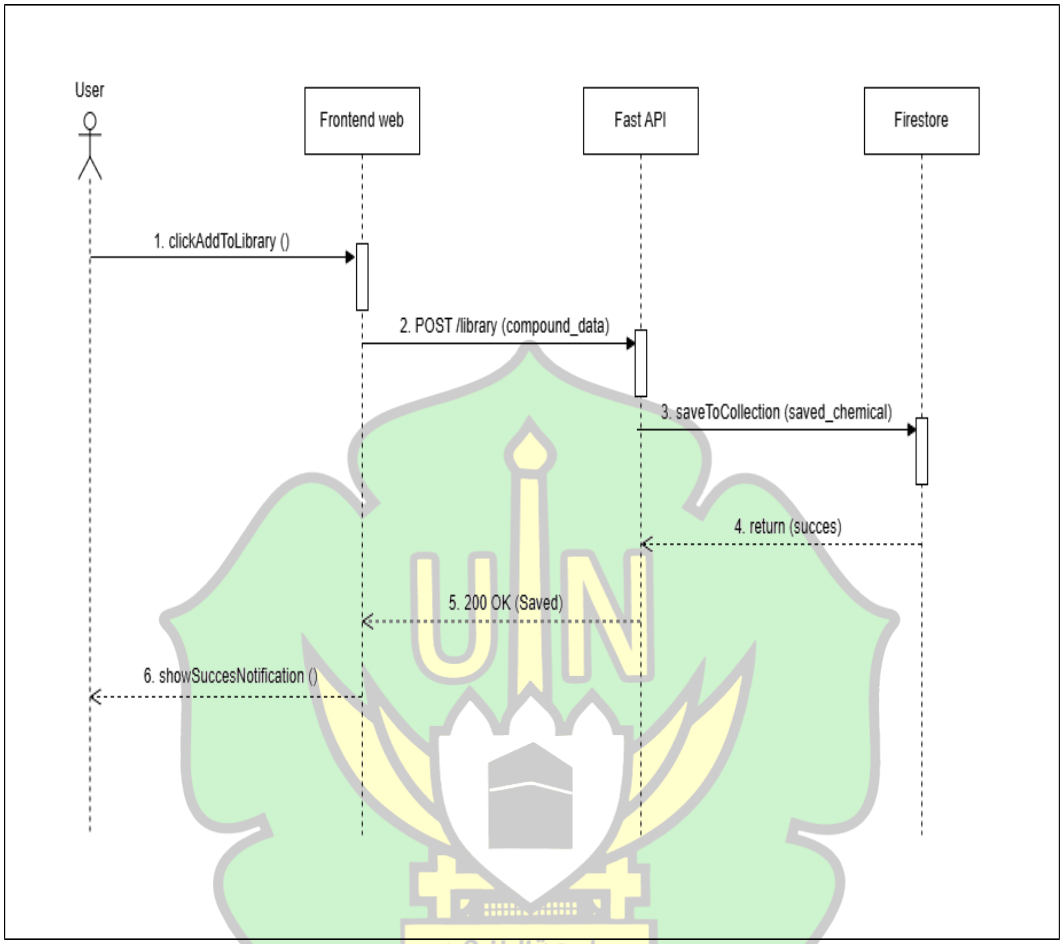


Gambar 3. 11 *Sequence Diagram* – Prediksi Senyawa & Auto-History

Gambar di atas mengilustrasikan urutan proses prediksi senyawa yang merupakan fungsi inti dari aplikasi web PolyChem. Alur dimulai saat pengguna menginput string SMILES, di mana FastAPI bertindak sebagai kontroler untuk memanggil modul RDKit guna ekstraksi fitur kimia. Secara paralel atau sekuensial, data tersebut dikirim ke Gemini API untuk menghasilkan analisis deskriptif. Sebagai bagian dari manajemen data otonom, sistem akan langsung menyimpan seluruh hasil komputasi ke dalam koleksi search_history di *database* Firestore sebelum mengirimkan respon akhir ke pengguna.

c. Sequence Diagram: Library Senyawa

Diagram ini menggambarkan fitur kurasi manual di mana pengguna dapat memilih senyawa tertentu untuk disimpan ke dalam koleksi pribadi.

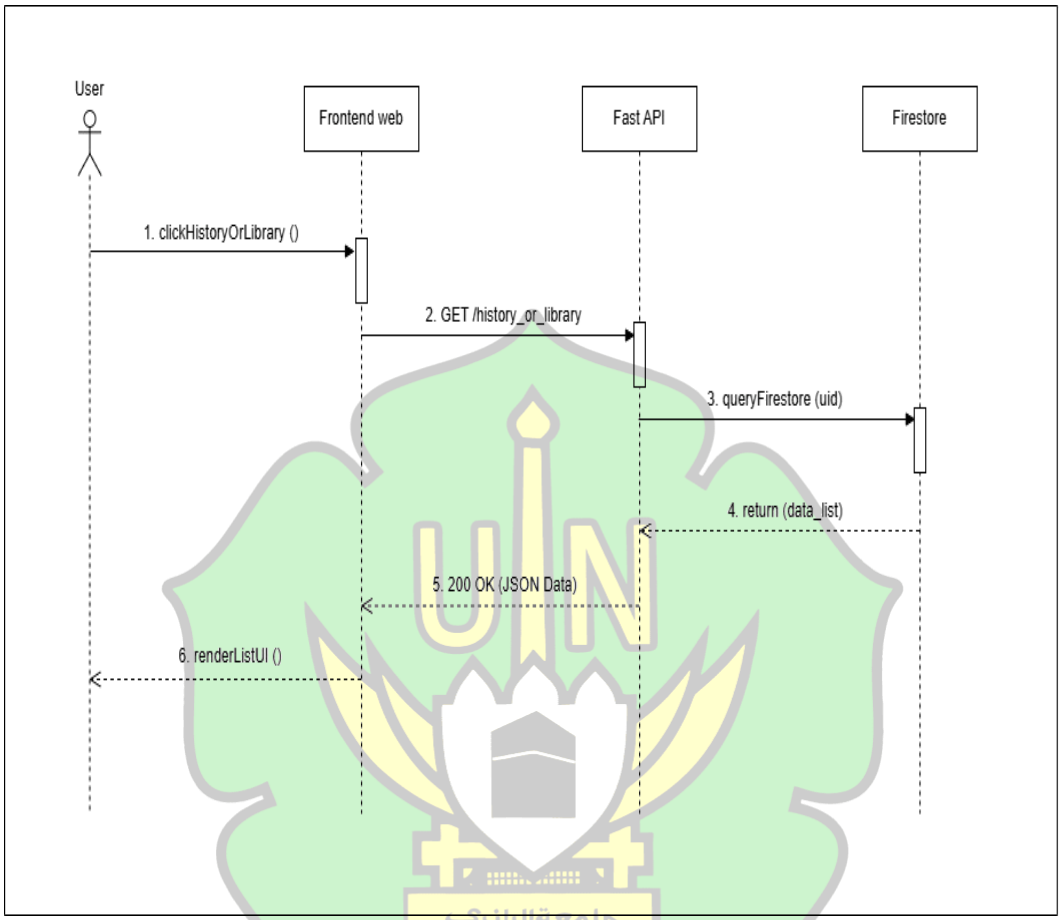


Gambar 3. 12 *Sequence Diagram* – Library Senyawa

Gambar di atas menjelaskan mekanisme penyimpanan senyawa ke dalam pustaka pribadi (Library). Berbeda dengan riwayat pencarian yang bersifat otomatis, alur ini dipicu secara manual melalui aksi pengguna yang menekan tombol "Add to Library" pada antarmuka hasil prediksi. FastAPI menerima permintaan tersebut dan melakukan operasi penyimpanan data senyawa spesifik ke dalam koleksi saved_chemicals pada Firestore, yang memungkinkan pengguna untuk mengakses kembali senyawa pilihan mereka di masa mendatang.

d. Sequence Diagram: Pengambilan Data (*Retrieval*)

Diagram ini menunjukkan bagaimana sistem memanggil kembali data yang telah disimpan untuk ditampilkan pada menu History dan Library.

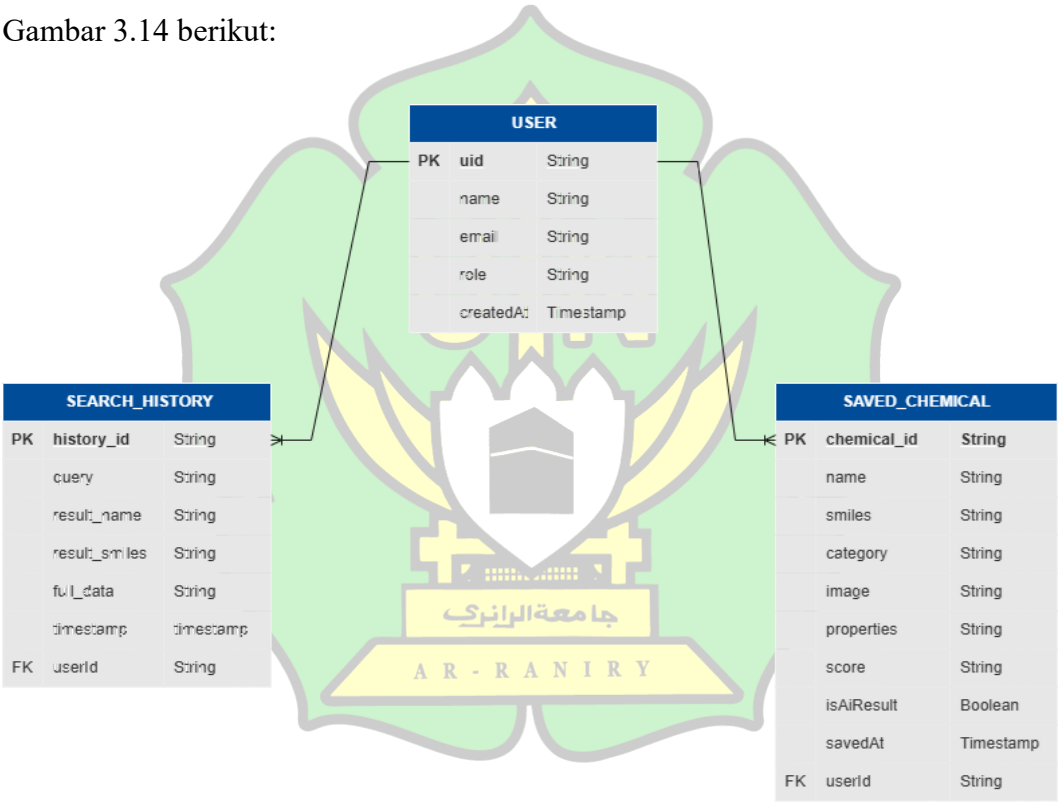


Gambar 3. 13 *Sequence Diagram* – Pengambilan Data

Gambar di atas merepresentasikan proses pengambilan data (data retrieval) untuk fitur riwayat dan pustaka senyawa. Saat pengguna mengakses menu terkait, *Frontend* Web mengirimkan permintaan HTTP GET kepada FastAPI yang menyertakan identitas unik pengguna (UID). Selanjutnya, *Backend* melakukan kueri ke koleksi Firestore yang relevan (baik search_history maupun saved_chemicals) untuk mengambil dokumen JSON yang tersimpan dan mengirimkannya kembali ke *Frontend* untuk dirender menjadi daftar informasi yang dapat dibaca oleh pengguna.

3.4.6 Perancangan Basisdata

Perancangan basis data pada aplikasi web PolyChem menggunakan layanan Firebase Firestore yang bersifat NoSQL. Meskipun demikian, struktur data tetap dirancang secara terorganisir untuk memudahkan pengelolaan data pengguna, riwayat prediksi, dan penyimpanan senyawa. Berdasarkan hasil perancangan *database* schema, terdapat tiga koleksi utama, yaitu users, search_history, dan saved_chemical. Relasi antar koleksi bersifat one-to-many, di mana satu pengguna dapat memiliki banyak data riwayat prediksi dan banyak data senyawa yang disimpan. Perancangan *database* schema aplikasi web PolyChem dapat dilihat pada Gambar 3.14 berikut:



Gambar 3. 14 *Database Schema* – aplikasi web PolyChem

Dari hasil perancangan *database* schema didapat tiga tabel dengan rincian yang dapat dilihat pada tabel dibawah.

Tabel 3. 5 Tabel – *User*

Field Name	***Data Type***	***Description***
uid	string	*Primary Key*
name	string	Nama pengguna
email	string	Email pengguna
role	string	Peran pengguna *(User)*
createdAt	timestamp	Waktu pembuatan akun

Tabel 3. 6 Tabel – *Search History*

Field Name	***Data Type***	***Description***
history_id	string	*Primary Key*
query	string	Input SMILES dari pengguna
result_name	string	Nama senyawa hasil prediksi
result_smiles	string	SMILES hasil prediksi
full_data	string	Data lengkap hasil prediksi (JSON)
timestamp	timestamp	Waktu prediksi dilakukan
userId	string	*Foreign Key*

Tabel 3. 7 Tabel – *Saved Chemical*

Field Name	***Data Type***	***Description***
chemical_id	string	*Primary Key*
name	string	Nama senyawa
smiles	string	Struktur SMILES
category	string	Kategori senyawa
image	string	URL gambar struktur senyawa

properties	string	Properti senyawa (JSON)
score	string	Nilai prediksi
isAiResult	boolean	Status hasil AI
savedAt	timestamp	Waktu penyimpanan
userId	string	*Foreign Key*

3.4.7 Perancangan Antarmuka (*User Interface Design*)

Perancangan antarmuka pengguna pada aplikasi PolyChem bertujuan untuk menyediakan tampilan yang sederhana, informatif, dan mudah dipahami. Desain UI/UX disusun untuk mendukung proses analisis senyawa polimer secara efisien, mulai dari input data hingga visualisasi hasil prediksi. Halaman-halaman utama yang dirancang pada aplikasi ini meliputi Home Screen, Output Screen, Library Screen, dan History Screen.

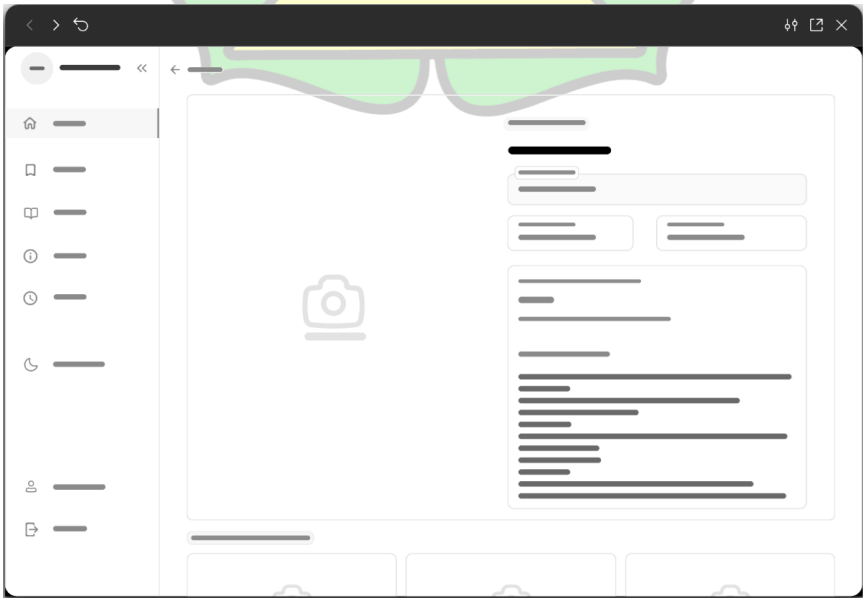
a. Home Screen

Home Screen merupakan halaman utama yang ditampilkan saat pengguna pertama kali mengakses aplikasi PolyChem. Halaman ini berfungsi sebagai titik awal interaksi pengguna dengan sistem serta memberikan gambaran singkat mengenai fungsi utama aplikasi sebagai *Plastic Discovery Agent*. Pada halaman ini, pengguna disajikan informasi identitas aplikasi dan navigasi menuju fitur utama, yaitu proses prediksi senyawa polimer. Desain antarmuka dibuat sederhana dan fokus pada kemudahan akses, sehingga pengguna dapat langsung melanjutkan ke proses analisis tanpa langkah yang kompleks. Home Screen dirancang untuk mendukung alur penggunaan yang cepat dan efisien sesuai dengan tujuan sistem.

Gambar 3. 15 Home Screen – *Wireframe*

b. Output Screen

Output Screen merupakan halaman yang menampilkan hasil prediksi senyawa polimer berdasarkan input SMILES yang diberikan oleh pengguna. Halaman ini menjadi fokus utama aplikasi karena menyajikan keluaran sistem berupa senyawa baru, nilai prediksi, serta justifikasi yang dihasilkan oleh model AI. Informasi ditampilkan secara terstruktur agar mudah dipahami, sehingga pengguna dapat langsung melihat hasil analisis tanpa perlu interpretasi tambahan. Desain halaman ini menekankan kejelasan data dan keterbacaan hasil prediksi untuk mendukung proses pengambilan keputusan pengguna.



Gambar 3. 16 Output Screen – *Wireframe*

c. Library Screen

Library Screen berfungsi sebagai halaman untuk menampilkan kumpulan data atau referensi senyawa polimer yang tersedia dalam sistem. Halaman ini memungkinkan pengguna untuk melihat daftar senyawa yang dapat digunakan sebagai acuan dalam proses analisis dan eksplorasi material. Tampilan dibuat dalam bentuk daftar atau kartu agar pengguna dapat dengan mudah menelusuri informasi yang tersedia. Library Screen dirancang untuk mendukung kebutuhan eksplorasi data secara sederhana dan terorganisir.

Gambar 3. 17 Library Screen – *Wireframe*

d. History Screen

History Screen merupakan halaman yang menampilkan riwayat prediksi yang telah dilakukan oleh pengguna sebelumnya. Halaman ini berguna untuk melacak hasil analisis terdahulu serta memudahkan pengguna dalam meninjau kembali output yang pernah dihasilkan sistem. Data riwayat disajikan secara ringkas dalam bentuk daftar agar pengguna dapat dengan cepat mengakses hasil prediksi tertentu. Dengan adanya halaman ini, aplikasi mendukung kontinuitas penggunaan dan dokumentasi hasil analisis pengguna.

Gambar 3. 18 History Screen – *Wireframe*

3.5 Evaluasi Prototype

Tahap ini merupakan tahap keempat dalam penelitian sesuai dengan alur pada Gambar 3.1, yaitu evaluasi prototype. Pada tahap ini dilakukan penilaian terhadap rancangan sistem yang telah dibuat pada tahap perancangan untuk memastikan kesesuaian antara kebutuhan pengguna dengan desain sistem yang dihasilkan. Evaluasi prototype dilakukan dengan cara meninjau kembali seluruh komponen perancangan sistem, meliputi arsitektur sistem, pemodelan UML, perancangan basis data, serta desain antarmuka pengguna. Proses evaluasi ini bertujuan untuk mengidentifikasi potensi kesalahan, kekurangan, maupun ketidaksesuaian rancangan sebelum sistem diimplementasikan.

Dalam proses evaluasi, dilakukan analisis terhadap alur sistem dan keterkaitan antar fitur berdasarkan skenario penggunaan (*user scenario*) yang telah didefinisikan sebelumnya melalui *user story*. Setiap fitur diperiksa apakah telah memenuhi kebutuhan pengguna serta berjalan sesuai dengan alur yang dirancang. Apabila ditemukan ketidaksesuaian antara rancangan dan kebutuhan pengguna, maka dilakukan perbaikan pada tahap perancangan sistem. Proses ini bersifat iteratif sesuai dengan metode prototyping, di mana perancangan dan evaluasi dilakukan secara berulang hingga diperoleh rancangan sistem yang optimal.

Hasil dari tahap evaluasi prototype ini menjadi dasar untuk melanjutkan ke tahap berikutnya, yaitu pengembangan sistem.

3.6 Pengembangan Sistem

Tahap pengembangan sistem merupakan tahap lanjutan setelah proses evaluasi prototype. Pada tahap ini, rancangan sistem yang telah divalidasi sebelumnya diimplementasikan menjadi sebuah aplikasi yang dapat dijalankan secara fungsional. Pengembangan dilakukan dengan mengintegrasikan seluruh komponen sistem, meliputi antarmuka pengguna (*frontend*), logika aplikasi (*backend*), basis data, serta layanan kecerdasan buatan ke dalam satu kesatuan sistem yang saling terhubung.

Pendekatan pengembangan sistem pada penelitian ini menggunakan arsitektur *client-server* yang terpisah (*decoupled architecture*), di mana *frontend* dan *backend* dikembangkan secara independen namun saling berkomunikasi melalui Application Programming Interface (API). Pendekatan ini dipilih untuk meningkatkan fleksibilitas, skalabilitas, serta kemudahan dalam proses pengembangan dan pemeliharaan sistem.

3.6.1 Spesifikasi Model dan Data

Aplikasi web PolyChem menerapkan arsitektur *Custom Agentic AI* yang dijalankan pada sisi *backend*. Logika agen dikembangkan menggunakan pendekatan berbasis fungsi (*function-based approach*) untuk mengorkestrasi interaksi antara model bahasa besar (*Large Language Model*), algoritma validasi kimia, serta basis data pengetahuan.

a. Large Language Model (LLM)

Untuk mengantisipasi keterbatasan kuota pada lapisan gratis (*free tier*) Gemini serta potensi gangguan layanan (*downtime*), arsitektur LLM pada *backend* aplikasi web PolyChem dirancang secara berlapis (*multi-layered*) guna menjamin ketersediaan sistem (*availability*) secara berkelanjutan. Sebelum permintaan diteruskan ke penyedia model AI mana pun, sistem terlebih dahulu memeriksa basis data *cache* lokal (*diskcache*); apabila notasi SMILES yang diminta pernah diproses

sebelumnya, hasil analisis (nama, justifikasi, dan nilai Tg) langsung dikembalikan tanpa memanggil API eksternal, sehingga mempercepat waktu respons secara signifikan. Apabila data belum tersedia di *cache*, permintaan diteruskan ke Gemini 2.5 Flash sebagai penyedia utama (*primary provider*) dengan strategi *fail-fast*: sistem membatasi waktu tunggu maksimum dan tidak melakukan pengulangan permintaan (*retry*) apabila terjadi kegagalan, seperti keterlambatan respons *server* atau kuota API yang telah habis, guna mencegah *bottleneck* pada sisi *backend*.

Jika Gemini gagal merespons dalam batas waktu tersebut, sistem secara otomatis mengalihkan permintaan ke penyedia cadangan (*fallback provider*), yaitu Groq dengan model llama-3.3-70b-versatile, tanpa menampilkan pesan kegagalan kepada pengguna. Groq kemudian mengambil alih pemrosesan permintaan yang sama, baik untuk penamaan senyawa, penyusunan justifikasi ilmiah, maupun prediksi nilai Tg. Dalam skenario ekstrem di mana kedua penyedia LLM tersebut tidak dapat diakses secara bersamaan, sistem tetap dapat beroperasi melalui mekanisme cadangan berbasis heuristik RDKit sebagaimana dijelaskan pada sub-bab Perancangan Pipeline Prediksi (BAB III). Dengan pendekatan berlapis ini, aplikasi web PolyChem tidak bergantung pada satu titik kegagalan (*single point of failure*) dan tetap responsif dalam skenario penggunaan berskala produksi.

b. Pemrosesan Molekuler

Validasi struktur kimia dilakukan menggunakan pustaka RDKit yang berjalan di lingkungan Python pada sisi *backend*. RDKit digunakan untuk memastikan bahwa input SMILES yang diberikan pengguna valid secara kimiawi sebelum diproses lebih lanjut. Selain itu, sistem juga memanfaatkan metode *Tanimoto Similarity* berbasis *Morgan Fingerprint* untuk mengukur tingkat kemiripan struktur molekul antara input pengguna dengan data referensi yang tersedia dalam basis data.

c. Dataset Referensi

Sistem menggunakan dataset sekunder yang bersumber dari repositori publik Kaggle dengan judul *"Extra dataset with SMILES, Tg, PID, Polymers Class"* dengan unique values 7178. Dataset ini digunakan sebagai basis

pengetahuan (*knowledge base*) dalam proses analisis dan pencocokan data. Dataset tersebut terdiri dari beberapa atribut penting, antara lain:

1. SMILES sebagai representasi struktur kimia
2. Tg (*Glass Transition Temperature*) sebagai parameter sifat termal
3. Polymer Class sebagai kategori material
4. PID sebagai identitas unik polimer

d. Preprocessing Data

Sebelum digunakan dalam sistem, dataset melalui tahap *preprocessing* untuk menjamin kualitas data. Tahapan ini meliputi validasi struktur SMILES menggunakan RDKit, penghapusan data yang memiliki nilai kosong (*missing values*), serta standarisasi format menjadi *canonical SMILES* untuk memastikan konsistensi dalam proses pencarian kemiripan.

3.6.2 Implementasi *Frontend*

Implementasi *frontend* pada aplikasi web PolyChem dilakukan menggunakan framework React.js yang berfungsi sebagai antarmuka pengguna (*user interface*). *Frontend* bertanggung jawab dalam menangani interaksi pengguna, seperti proses input data SMILES, navigasi antar halaman, serta penyajian hasil prediksi. Pengembangan *frontend* dilakukan dengan pendekatan berbasis komponen (*component-based architecture*), sehingga setiap bagian antarmuka dikembangkan secara modular dan terstruktur. Hal ini memudahkan dalam pengelolaan kode serta meningkatkan skalabilitas sistem. Komunikasi antara *frontend* dan *backend* dilakukan melalui API berbasis HTTP menggunakan metode *request-response*. *Frontend* mengirimkan data input pengguna ke *backend* dalam format JSON, kemudian menerima hasil pemrosesan untuk ditampilkan kembali kepada pengguna. Selain itu, *frontend* juga mengelola *state* aplikasi untuk menjaga konsistensi data selama interaksi berlangsung.

3.6.3 Implementasi *Backend*

Backend pada aplikasi web PolyChem dikembangkan menggunakan framework FastAPI yang berfungsi sebagai pengelola logika bisnis dan pemrosesan data. *Backend* bertanggung jawab dalam menerima permintaan dari *frontend*, melakukan validasi data, serta mengelola proses integrasi dengan layanan AI dan basis data. Pada fitur prediksi, *backend* melakukan validasi terhadap input SMILES menggunakan RDKit. Setelah input dinyatakan valid, data akan diproses lebih lanjut dengan memanfaatkan model kecerdasan buatan untuk menghasilkan prediksi properti polimer. *Backend* juga terintegrasi dengan layanan Firebase Firestore sebagai basis data untuk menyimpan informasi pengguna, riwayat prediksi, serta data koleksi senyawa. Seluruh komunikasi antara *frontend* dan *backend* diatur melalui *endpoint* API yang dirancang secara sistematis untuk mendukung efisiensi dan keamanan pertukaran data.

3.6.4 Integrasi Sistem

Integrasi sistem dilakukan untuk menghubungkan seluruh komponen yang telah dikembangkan agar dapat bekerja secara terpadu. Proses integrasi mencakup komunikasi antara *frontend*, *backend*, model kecerdasan buatan, serta basis data dalam satu alur sistem yang terstruktur. Alur integrasi dimulai dari pengguna yang memasukkan data SMILES melalui antarmuka *frontend*. Data tersebut kemudian dikirimkan ke *backend* melalui API untuk dilakukan proses validasi menggunakan RDKit. Setelah dinyatakan valid, data diteruskan ke layanan AI untuk menghasilkan prediksi properti polimer.

Hasil prediksi yang diperoleh kemudian disimpan ke dalam basis data Firebase sebagai bagian dari riwayat pengguna. Selanjutnya, *backend* mengirimkan hasil tersebut kembali ke *frontend* untuk ditampilkan kepada pengguna dalam bentuk informasi yang terstruktur dan mudah dipahami. Pendekatan integrasi ini tidak hanya memastikan kelancaran alur data dari input hingga output, tetapi juga membantu meminimalisir kesalahan akibat keterbatasan model AI dengan mengombinasikan validasi berbasis algoritma kimia. Dengan demikian, aplikasi

web PolyChem mampu menghasilkan prediksi yang lebih akurat, terstruktur, dan dapat diandalkan.

3.7 Pengujian Sistem

Tahap pengujian sistem merupakan tahap yang dilakukan setelah proses pengembangan sistem selesai. Pada tahap ini dilakukan pengujian terhadap seluruh fitur yang telah diimplementasikan untuk memastikan sistem berjalan sesuai dengan kebutuhan yang telah ditentukan. Metode pengujian yang digunakan dalam penelitian ini adalah *black-box testing*, yaitu metode pengujian yang berfokus pada fungsi sistem tanpa melihat struktur kode program. Pengujian dilakukan dengan memberikan input tertentu pada sistem dan mengamati apakah output yang dihasilkan sesuai dengan yang diharapkan.

Pengujian dilakukan pada fitur-fitur utama dalam aplikasi PolyChem, antara lain:

1. Fitur Autentikasi

   Pengujian dilakukan untuk memastikan proses login dan registrasi berjalan dengan benar serta sistem mampu memvalidasi data pengguna.
2. Fitur Prediksi Polimer

   Pengujian dilakukan dengan memasukkan data SMILES untuk memastikan sistem dapat memproses input, melakukan validasi menggunakan RDKit, serta menghasilkan output prediksi melalui model AI.
3. Fitur History

   Pengujian dilakukan untuk memastikan sistem dapat menyimpan dan menampilkan riwayat prediksi pengguna dengan benar.
4. Fitur Library

   Pengujian dilakukan untuk memastikan pengguna dapat menyimpan dan mengakses data senyawa yang telah dipilih.

Hasil dari pengujian ini digunakan untuk mengetahui apakah sistem telah berjalan sesuai dengan yang diharapkan atau masih terdapat kesalahan yang perlu diperbaiki.

3.8 Evaluasi Sistem

Tahap evaluasi sistem merupakan tahap lanjutan setelah proses pengujian sistem yang bertujuan untuk menilai kinerja serta kesesuaian aplikasi web PolyChem secara menyeluruh terhadap kebutuhan pengguna dan tujuan penelitian. Evaluasi ini dilakukan dengan menganalisis hasil pengujian yang telah diperoleh pada tahap sebelumnya, sehingga dapat diketahui sejauh mana sistem mampu menjalankan fungsi-fungsi yang telah dirancang.

Proses evaluasi difokuskan pada aspek fungsionalitas, keandalan sistem, serta kesesuaian hasil yang dihasilkan oleh sistem dengan kebutuhan pengguna. Selain itu, evaluasi juga mempertimbangkan kemudahan penggunaan (usability) dari antarmuka yang telah dirancang, sehingga sistem tidak hanya berjalan dengan baik secara teknis, tetapi juga dapat digunakan secara efektif oleh pengguna. Apabila dalam proses evaluasi ditemukan kesalahan, ketidaksesuaian, atau kekurangan pada sistem, maka dilakukan proses perbaikan dan penyempurnaan pada tahap pengembangan. Proses ini bersifat iteratif, sesuai dengan pendekatan prototyping yang digunakan dalam penelitian, di mana sistem dapat mengalami beberapa kali siklus perbaikan hingga mencapai kondisi yang optimal.

Hasil dari evaluasi sistem ini menjadi dasar dalam menentukan tingkat keberhasilan sistem yang telah dikembangkan. Apabila sistem telah memenuhi seluruh kebutuhan fungsional dan non-fungsional serta tidak ditemukan kesalahan yang signifikan, maka sistem dinyatakan layak untuk digunakan. Dengan demikian, tahap evaluasi sistem ini berperan penting dalam memastikan bahwa aplikasi PolyChem mampu memberikan solusi yang sesuai dengan tujuan penelitian, yaitu menyediakan sistem prediksi properti polimer yang akurat, efisien, dan mudah digunakan.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Implementasi dan Integrasi Sistem

Implementasi sistem merupakan tahap realisasi dari rancangan yang telah disusun pada Bab III ke dalam bentuk aplikasi PolyChem berbasis web. Pada tahap ini, seluruh komponen sistem diimplementasikan menggunakan arsitektur client-server yang memisahkan sisi *frontend* dan *backend*, serta diintegrasikan dengan layanan pendukung seperti Firebase, RDKit, dan Google Gemini API sehingga seluruh fitur yang telah dirancang dapat berjalan sesuai dengan kebutuhan pengguna.

4.1.1 Implementasi Basis Data

Implementasi basis data pada aplikasi web PolyChem menggunakan layanan *Cloud Firestore* (NoSQL) dari Firebase. Untuk menjamin keamanan dan integritas data, aplikasi ini menerapkan arsitektur *Backend-Mediated Database*. Dengan pendekatan ini, antarmuka *frontend* (React.js) tidak memiliki akses langsung untuk memanipulasi *database*. Seluruh operasi *Input/Output* (CRUD) dikendalikan secara mutlak oleh *server backend* (FastAPI) menggunakan Firebase Admin SDK. *Frontend* hanya bertugas mengirimkan token identitas (*ID Token*), sementara *server* akan memverifikasi token tersebut sebelum mengizinkan akses ke koleksi spesifik.

1. Konfigurasi dan Inisialisasi *Firebase Admin SDK*

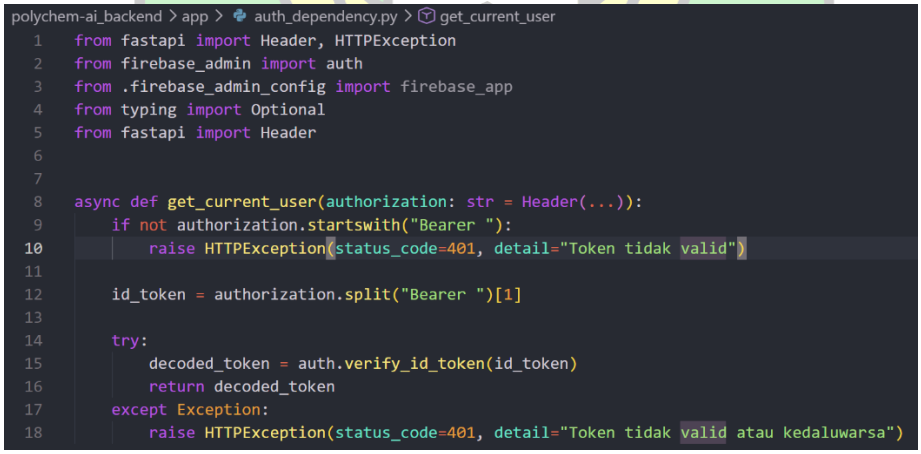
Langkah pertama dalam implementasi ini adalah menghubungkan *server* Python dengan proyek Firebase menggunakan berkas kredensial (*Service Account Key*). Berkas kredensial ini menjamin bahwa *server* memiliki otoritas penuh sebagai administrator basis data. Potongan kode inisialisasi pada sisi *server* ditunjukkan pada Gambar 4.1 berikut.

Gambar 4. 1 Inisialisasi dan Konfigurasi *Firebase Admin SDK*

Pada Gambar 4.1, pustaka firebase_admin diinisialisasi menggunakan credentials.Certificate(). Setelah koneksi berhasil terjalin, objek antarmuka basis data dideklarasikan ke dalam variabel db = firestore.client(). Variabel inilah yang nantinya akan diimpor oleh seluruh modul layanan untuk melakukan manipulasi koleksi.

2. Implementasi Verifikasi Sesi *(Authentication Dependency)*

Sebagai gerbang keamanan utama, setiap *endpoint* API yang akan mengakses basis data dilindungi oleh fungsi injeksi dependensi (*dependency injection*). Potongan kode yang bertugas melakukan verifikasi token ini disajikan pada Gambar 4.2 berikut.

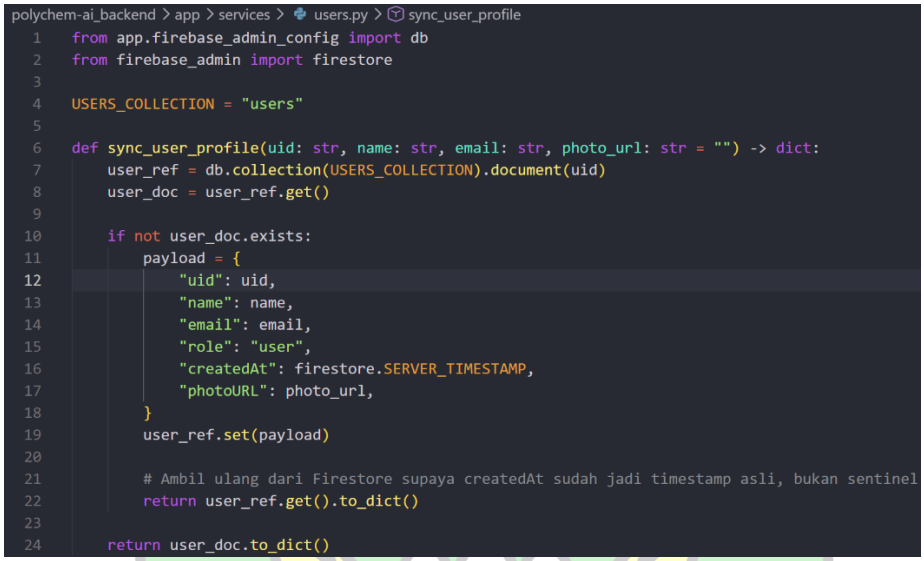


Gambar 4. 2 Implementasi Verifikasi *ID Token* Firebase pada *server*

Berdasarkan Gambar 4.2, fungsi get_current_user bertugas mengekstraksi token dari *header* HTTP Authorization: Bearer. Token tersebut kemudian diverifikasi secara kriptografis menggunakan auth.verify_id_token(id_token). Jika valid, sistem akan mengekstraksi parameter uid (*User ID*) untuk memastikan bahwa pengguna hanya dapat memanipulasi data yang menjadi miliknya sendiri (*Tenant Isolation*).

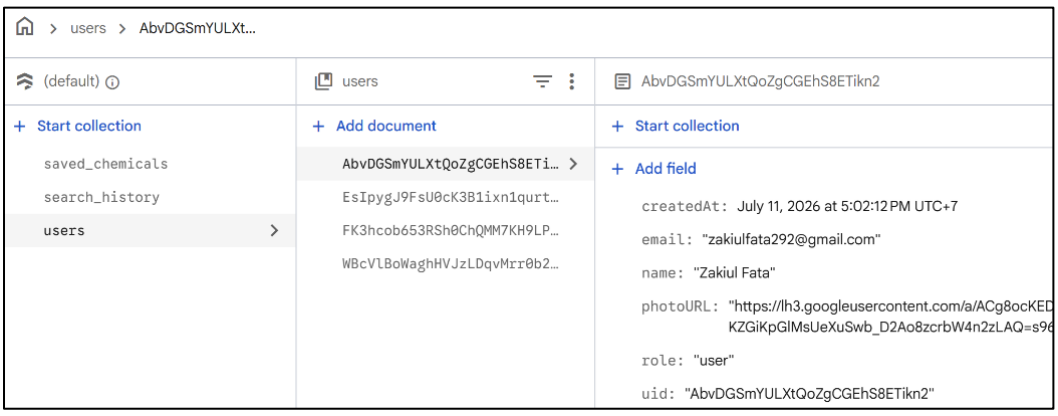
3. Implementasi Sinkronisasi Koleksi Pengguna *(Users)*

Ketika pengguna berhasil melakukan registrasi atau *login* melalui Google SSO di *frontend*, antarmuka akan memanggil *endpoint* POST /auth/sync-profile. *server* kemudian menangkap data tersebut dan menyimpannya ke dalam koleksi users. Potongan kode yang mengeksekusi integrasi ini ditunjukkan pada Gambar 4.3 berikut.



Gambar 4. 3 Potongan Kode Sinkronisasi Profil Pengguna ke Firestore

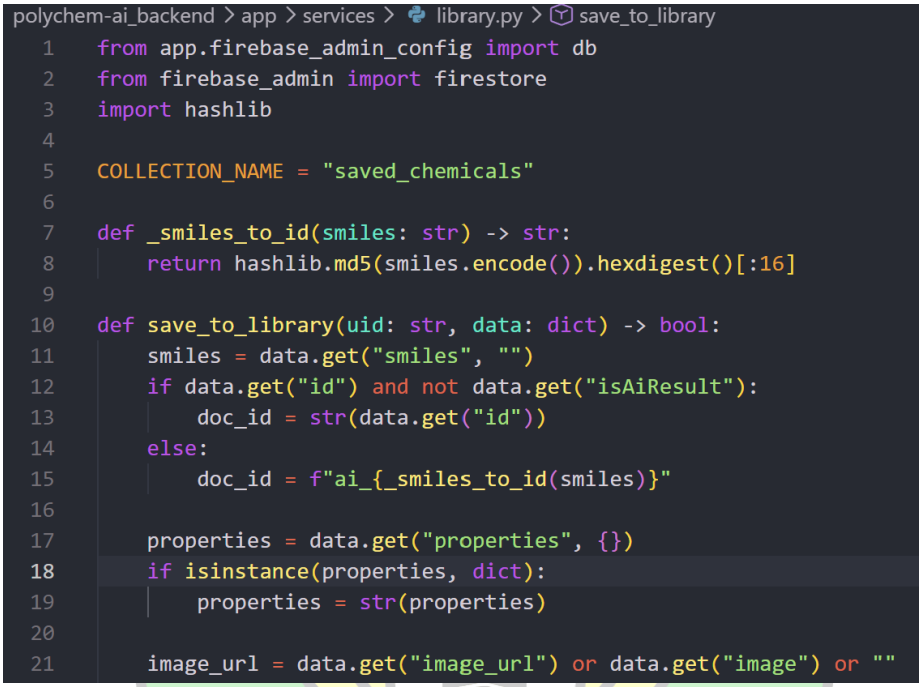
Pada Gambar 4.3, fungsi sync_user_profile memeriksa eksistensi dokumen pengguna menggunakan uid. Jika belum ada, sistem akan membuat dokumen baru dengan ID yang sinkron dengan Firebase Auth menggunakan metode set(). Keberhasilan proses sinkronisasi ini dapat diverifikasi melalui data profil di *Firebase Console* sebagaimana ditunjukkan pada Gambar 4.4.



Gambar 4. 4 Tampilan Koleksi *Users* pada *Firebase Console*

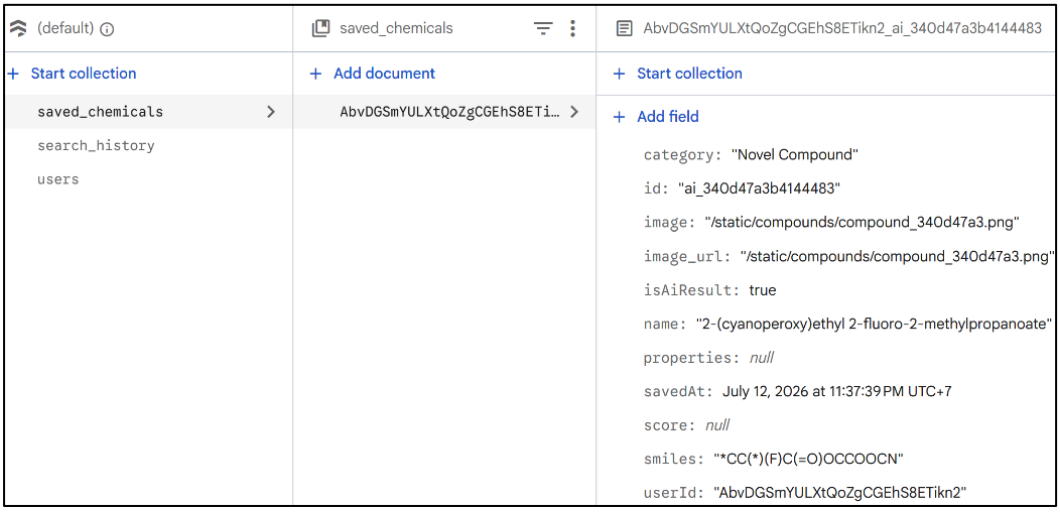
4. Implementasi Koleksi Pustaka Senyawa *(Saved Chemicals)*

Fitur *Library* memungkinkan peneliti untuk menyortir dan menyimpan hasil prediksi AI yang dianggap relevan. Proses penyimpanan ini ditangani oleh fungsi save_to_library. Potongan kode yang mengeksekusi penyimpanan kurasi senyawa ini disajikan pada Gambar 4.5 berikut.



Gambar 4. 5 Potongan Kode Penyimpanan Senyawa ke Koleksi *Saved Chemicals*

Pada Gambar 4.5, sistem mencegah tabrakan data (*data collision*) dan replikasi antar-pengguna dengan merangkai identitas unik berjenis *Composite ID* (f"{uid}_{doc_id}"). Jika pengguna ingin menyimpan senyawa berulang kali, *Firestore* hanya akan memperbarui dokumen yang sama tanpa membuat entri ganda, sehingga efisiensi kapasitas penyimpanan tetap terjaga. Hasil penyimpanan tersebut dapat diverifikasi melalui *Firebase Console* sebagaimana ditunjukkan pada Gambar 4.6.



Gambar 4. 6 Tampilan Koleksi *Saved Chemicals* pada *Firebase Console*

5. Implementasi Koleksi Riwayat Otomatis *(Search History)*

Berbeda dengan fitur *Library* yang disimpan secara manual oleh pengguna, jejak prediksi riset direkam secara otonom oleh sistem di latar belakang (*background task*). Fungsi ini disematkan secara terintegrasi di dalam *endpoint* /predict.

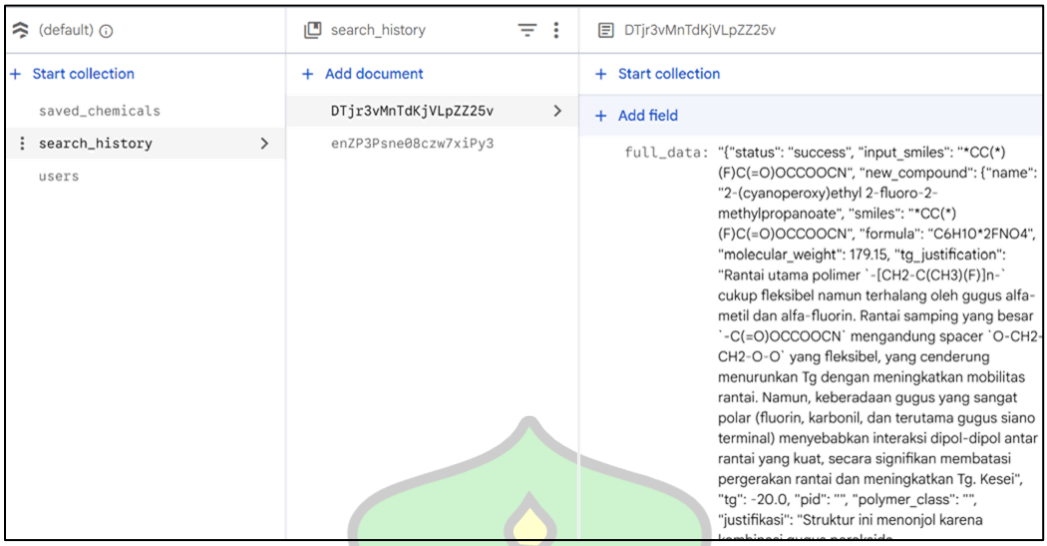
polychem-ai_backend > app > services > history.py > get_user_history > limit_count

```python
def add_to_history(uid: str, query_text: str, result_data: dict):
    nc = result_data.get("new_compound", {})
    payload = {
        "userId": uid,
        "query": query_text,
        "result_name": nc.get("name", "Unknown"),
        "result_smiles": nc.get("smiles", ""),
        "full_data": json.dumps(result_data),
        "timestamp": firestore.SERVER_TIMESTAMP,
    }
    db.collection(HISTORY_COLLECTION).add(payload)
```

Gambar 4. 7 Potongan Kode Pencatatan Riwayat Prediksi Otomatis

Berdasarkan Gambar 4.7, sistem menggunakan fungsi db.collection(HISTORY_COLLECTION).add(payload). Pendekatan add() dipilih secara spesifik karena *server Firestore* akan menghasilkan (*generate*) ID dokumen secara acak, yang sangat ideal untuk menampung riwayat di mana satu pengguna dipastikan akan memiliki puluhan atau ratusan entri aktivitas log riset. Mengingat struktur data keluaran dari *Agentic AI* sangat kompleks, atribut utama dikonversi menjadi tipe data teks menggunakan json.dumps(result_data). Hasil dari pencatatan

otomatis ini dapat ditinjau pada *Firebase Console* sebagaimana ditunjukkan pada Gambar 4.8.



Gambar 4. 8 Tampilan Koleksi *Search History* pada *Firebase Console*

6. Operasi Pengambilan dan Penghapusan Data

Untuk menyajikan data kembali ke peramban pengguna, *server* menyediakan mekanisme operasional untuk mengambil (*retrieve*) dan menghapus (*delete*) data. Potongan kode untuk fungsi pengambilan riwayat disajikan pada Gambar 4.9 berikut.

polychem-ai_backend > app > services > history.py > get_user_history

```
20  def get_user_history(uid: str, limit_count: int = 50) -> list:
21      docs = (
22          db.collection(HISTORY_COLLECTION)
23          .where("userId", "==", uid)
24          .order_by("timestamp", direction=firestore.Query.DESCENDING)
25          .limit(limit_count)
26          .stream()
27      )
28      items = []
29      for doc in docs:
30          d = doc.to_dict()
31          d["id"] = doc.id
32          items.append(d)
33      return items
```

Gambar 4. 9 Potongan Kode Pengambilan Data dengan Filter dan *Ordering*

Pada Gambar 4.9, fungsi get_user_history mengamankan data pengguna melalui filter operasional .where("userId", "==", uid). Untuk memastikan antarmuka *History* menampilkan jejak penelusuran terbaru di urutan teratas, kueri tersebut dipadukan dengan metode pengurutan .order_by("timestamp",

direction=firestore.Query.DESCENDING) serta pembatasan jumlah (*limit*) untuk menghemat laju *bandwidth*.

7. Ringkasan Struktur Basis Data

Sebagai kesimpulan dari seluruh implementasi operasi basis data di atas, aplikasi web PolyChem telah berhasil memetakan struktur tipe data yang konsisten. Ringkasan akhir dari struktur basis data NoSQL yang terbentuk disajikan pada Tabel 4.1 berikut.

Tabel 4. 1 Ringkasan Struktur Koleksi pada Basis Data PolyChem

Nama Koleksi	Tipe ID Dokumen	Atribut (*Field*) Utama	Deskripsi
users	Sesuai uid Firebase *Auth*	uid, name, email, role, createdAt	Profil pengguna sebagai relasi utama.
saved_chemicals	userId + ID Senyawa	smiles, properties, score, isAiResult	Pustaka senyawa (*Library*) simpanan manual.
search_history	Acak (Otomatis)	query, result_name, full_data, timestamp	Rekaman log prediksi dari *Agentic AI*.

4.1.2 Implementasi *Backend* (FastAPI)

Implementasi *backend* pada aplikasi web PolyChem menggunakan *framework* FastAPI, sebuah *framework* web modern berbasis Python yang dirancang untuk performa tinggi dan pengembangan yang cepat. *Backend* ini bertindak sebagai pusat orkestrasi yang menghubungkan logika antarmuka (*frontend*), pemrosesan data kimia (*RDKit*), serta kecerdasan buatan (*Google Gemini API dan Groq API*).

1. Konfigurasi Lingkungan dan *Middleware* (FastAPI)

Tahap awal dalam implementasi *backend* adalah inisialisasi aplikasi dan pengaturan kebijakan akses antar-sistem (*Cross-Origin Resource Sharing* atau CORS). Pengaturan ini sangat krusial agar aplikasi *frontend* berbasis React dapat berkomunikasi dengan *server* FastAPI dengan aman tanpa terblokir oleh kebijakan keamanan *browser*.

Potongan kode inisialisasi aplikasi dan pengaturan *middleware* disajikan pada Gambar 4.10 berikut.

```python
from contextlib import asynccontextmanager
import os
from rdkit import RDLogger
RDLogger.DisableLog("rdApp.*")
from fastapi import FastAPI, HTTPException
from fastapi.middleware.cors import CORSMiddleware
from fastapi.staticfiles import StaticFiles
from fastapi.encoders import jsonable_encoder
from fastapi.responses import JSONResponse
from app.schemas import RecommendRequest, RecommendResponse
from app.store import load_dataset
from app.core import recommend_new_compound
from app.settings import STATIC_DIR, COMPOUNDS_DIR
from app.cache import get_history

@asynccontextmanager
async def lifespan(app: FastAPI):
    print("Loading dataset...")
    load_dataset()
    print("Dataset loaded!")
    yield

app = FastAPI(
    title="Novachem AI Backend",
    version="1.0.0",
    lifespan=lifespan,
)
```

Gambar 4. 10 Konfigurasi Inisialisasi FastAPI dan *Middleware* CORS

Berdasarkan potongan kode pada Gambar 4.10, inisialisasi aplikasi dilakukan melalui fungsi app = FastAPI(...) yang memuat konfigurasi metadata sistem. Salah satu elemen terpenting dalam kode tersebut adalah fungsi lifespan(app: FastAPI). Fungsi ini menggunakan dekorator @asynccontextmanager

untuk melakukan pemuatan *dataset* polimer (load_dataset()) hanya sekali saat *server* pertama kali dijalankan. Pendekatan ini memastikan efisiensi memori karena *dataset* tidak perlu dimuat ulang pada setiap *request* yang masuk.

Selain itu, sistem mengimplementasikan *CORS Middleware* untuk mengatur kebijakan akses API. Dalam tahap pengembangan, parameter allow_origins = ["*"] digunakan untuk memfasilitasi komunikasi antara *server* lokal dengan *client* React. Penggunaan *middleware* ini memungkinkan sistem untuk mengelola sesi komunikasi *request-response* dengan aman antara sisi *frontend* dan *backend*, yang menjadi fondasi utama sebelum aplikasi dapat menerima input dari pengguna.

2. Implementasi Pemrosesan Senyawa Kimia (RDKit)

Proses pemrosesan data kimia di sisi *backend* dilakukan menggunakan pustaka RDKit. Modul ini bertanggung jawab untuk mengubah input teks *SMILES* menjadi representasi digital yang dapat diolah oleh mesin, serta melakukan analisis komputasi untuk mendukung pengambilan keputusan oleh AI.

Potongan kode yang menangani validasi molekuler dan ekstraksi fitur kimia disajikan pada Gambar 4.11 berikut.

```python
def find_similar_compounds(input_fp_rdkit, compound_name: str, top_k: int = 3):
        if store.df is None or store.dataset_rdkit_fps is None:
            return []

        similarities = DataStructs.BulkTanimotoSimilarity(
            input_fp_rdkit, store.dataset_rdkit_fps
        )
        similarities = np.array(similarities, dtype=float)

        top_indices = np.argsort(similarities)[::-1][:top_k]
        top_smiles = tuple(str(store.df.iloc[idx]["SMILES"]).strip() for idx in top_indices)

        justifs = _cached_similar_justifs(compound_name, top_smiles)
        if not isinstance(justifs, list):
            justifs = ["Mirip secara struktur umum (fallback)."] * len(top_smiles)

        similar_compounds = []
```

Gambar 4. 11 Modul Pemrosesan Struktur Molekul dengan RDKit

Berdasarkan Gambar 4.11, fungsi build_fingerprints() melakukan konversi struktur molekul menjadi *Morgan Fingerprint*. *Fingerprint* ini adalah representasi biner yang menggambarkan lingkungan atom dalam molekul, yang memungkinkan sistem untuk melakukan perhitungan kemiripan struktural secara efisien.

Selain validasi, sistem menerapkan fungsi find_similar_compounds() yang mengintegrasikan metode *Tanimoto Similarity*. Metode ini digunakan untuk mengukur tingkat kesamaan antara molekul input dengan *dataset* referensi polimer. Sistem akan mencari *top-k* senyawa dengan nilai kemiripan tertinggi, yang kemudian diperkaya dengan justifikasi berbasis AI. Jika data sifat termal (*Glass Transition Temperature*) tidak tersedia dalam basis data, sistem secara otomatis menjalankan fungsi _tg_fallback_local() sebagai mekanisme cadangan untuk melakukan estimasi berdasarkan atribut kimia seperti jumlah cincin, ikatan rotasi, serta berat molekul (*Molecular Weight*).

Integrasi ini menunjukkan bahwa sistem tidak hanya mengandalkan AI generatif, tetapi juga menerapkan validasi berbasis algoritma kimia yang ketat (pendekatan *neuro-symbolic*) untuk memastikan hasil prediksi memiliki landasan saintifik.

3. Integrasi *Agentic AI*

Aplikasi web PolyChem mengadopsi arsitektur *Agentic AI* yang bersifat *neuro-symbolic*, di mana integrasi antara model komputasi kimia mandiri dengan *Large Language Model* (LLM) menjadi inti dari proses penalaran (*reasoning*). Dalam arsitektur ini, sistem memanfaatkan pustaka RDKit untuk validasi struktur dan perhitungan kimia, sementara Google Gemini 2.5 Flash dan Groq Llama 3.3 berperan secara bergantian sebagai *Justification Agent* yang menginterpretasikan hasil komputasi menjadi narasi ilmiah yang logis. Modul LLM dirancang untuk membungkus eksekusi agen utama dan agen cadangan di dalam struktur penanganan eksepsi berantai (*safe_invoke*), guna mencegah terjadinya titik kegagalan tunggal (*single point of failure*) pada saat melakukan penalaran (*reasoning*).

Potongan kode yang menunjukkan struktur dependensi modul dan logika pemrosesan kimia disajikan pada Gambar 4.12 berikut.

```python
import numpy as np
from functools import lru_cache
from typing import List, Tuple
from rdkit import Chem, DataStructs
from rdkit.Chem import AllChem, Descriptors, rdMolDescriptors
import app.store as store
from app.llm import (
    generate_compound_name,
    generate_new_justification,
    justify_similar_compounds_batch,
    predict_tg_with_llm,
)

try:
    from app.llm import tg_fallback_heuristic as _tg_fallback_heuristic
except Exception:
    _tg_fallback_heuristic = None

from app.images import save_smiles_png
from app.cache import cache, key_new_compound, key_similar_justif, push_history

from app.settings import COMPOUNDS_DIR

```

Gambar 4. 12 Struktur Dependensi Modul *Backend*

Berdasarkan Gambar 4.12, sistem menerapkan pendekatan pengembangan *modular*. Modul app.llm diimpor untuk menangani komunikasi dengan agen AI, sementara pustaka RDKit (seperti DataStructs, AllChem) digunakan untuk ekstraksi fitur kimia. Pemisahan modul ini menjamin bahwa setiap lapisan sistem, mulai dari pemrosesan data hingga integrasi AI, dapat dikelola dan dikembangkan secara independen.

4. Implementasi *Endpoint* API

Server FastAPI pada aplikasi web PolyChem menyediakan berbagai rute (*endpoint*) API yang berfungsi sebagai jembatan komunikasi data dengan antarmuka klien. Setiap *endpoint* menangani fungsi spesifik dan sebagian besar dilindungi oleh mekanisme verifikasi token untuk menjamin keamanan data. Berikut adalah penjabaran dari masing-masing *endpoint* yang diimplementasikan:

a. *Endpoint* Pengecekan Status *server* (*GET /health*)

Endpoint ini merupakan rute fundamental yang diimplementasikan untuk melakukan pengecekan ketersediaan layanan (*health check*).

```
# ============================================================
# API Endpoints
# ============================================================

@app.get("/health")
def health():
    return {"status": "ok"}
```

Gambar 4. 13 Implementasi *Endpoint* Pengecekan Status

Pada Gambar 4.13 Fungsi health() bekerja menggunakan metode GET dan hanya mengembalikan objek JSON sederhana {"status": "ok"}. *Endpoint* ini memastikan bahwa *server* telah beroperasi dengan baik dan siap menerima permintaan masuk sebelum antarmuka klien melakukan pemanggilan ke fungsi yang lebih kompleks.

b. *Endpoint* Prediksi Senyawa *(POST /predict)*

Endpoint ini merupakan inti pemrosesan komputasi dari aplikasi PolyChem yang menghubungkan input pengguna dengan algoritma *RDKit* dan *Agentic AI*. Potongan kode yang mengeksekusi proses ini ditunjukkan pada Gambar 4.14 berikut.

```
@app.post("/predict", response_model=RecommendResponse)
def predict(req: RecommendRequest, user=Depends(get_optional_user)):
    try:
        result = recommend_new_compound(req.smiles)
    except Exception as e:
        print(f"Error di pipeline: {e}")
        raise HTTPException(status_code=503, detail=f"Pipeline error: {str(e)}")

    if "error" in result:
        raise HTTPException(status_code=400, detail=result["error"])

    if user:
        try:
            add_to_history(user["uid"], req.smiles, result)
        except Exception as e:
            print(f"⚠️ Gagal simpan history: {e}")

    return result
```

Gambar 4. 14 Implementasi *Endpoint* Prediksi dan Riwayat Otomatis

Berdasarkan Gambar 4.14, *endpoint* ini menggunakan metode POST dan menerima muatan data (*payload*) berdasarkan skema RecommendRequest. Fungsi ini mengeksekusi *pipeline* melalui modul recommend_new_compound(). Jika terjadi galat pemrosesan (misalnya *SMILES* tidak valid), sistem akan membangkitkan HTTPException dengan kode 400 atau 503 untuk mencegah *crash*.

Keunggulan utama pada *endpoint* ini adalah penggunaan injeksi dependensi get_optional_user. Jika permintaan dikirim dengan *ID Token* yang sah, sistem akan secara otomatis memanggil add_to_history() untuk mencatat jejak riset tersebut ke dalam basis data *Firestore*.

c. *Endpoint* Sinkronisasi Profil Pengguna *(POST /auth/sync-profile)*

Endpoint ini bertugas untuk memastikan data profil dari penyedia otorisasi pihak ketiga (*Google SSO*) tersinkronisasi ke dalam *database* sistem, sebagaimana ditunjukkan pada Gambar 4.15 berikut.

```python
@app.post("/auth/sync-profile")
def sync_profile(req: SyncProfileRequest, user=Depends(get_current_user)):
    uid = user["uid"]
    email = user.get("email", "")
    name = req.name or user.get("name", "Unknown")
    try:
        profile = sync_user_profile(uid, name, email, req.photo_url or "")
        return {"success": True, "profile": profile}
    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Gagal sinkronisasi profil: {str(e)}")
```

Gambar 4. 15 Implementasi *Endpoint* Sinkronisasi Profil

Berdasarkan Gambar 4.15, fungsi sync_profile mewajibkan adanya token yang valid melalui dependensi get_current_user. Sistem kemudian mengekstraksi informasi uid, email, dan name dari token tersebut, lalu memanggil fungsi sync_user_profile() untuk mencatat atau memperbarui dokumen pengguna di koleksi users pada *Firestore*.

d. *Endpoint* Penyimpanan Pustaka Senyawa *(POST /library)*

Fitur penyimpanan ke *Library* dikelola melalui *endpoint* terproteksi yang memastikan senyawa riset hanya disimpan ke dalam profil pengguna yang memintanya. Potongan kode *endpoint* ini ditunjukkan pada Gambar 4.16 berikut.

```python
@app.post("/library")
def add_to_library(req: SaveLibraryRequest, user=Depends(get_current_user)):
    uid = user["uid"]
    try:
        save_to_library(uid, req.dict())
        return {"success": True}
    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Gagal menyimpan: {str(e)}")
```

Gambar 4. 16 Implementasi *Endpoint* Penyimpanan Senyawa ke Pustaka

Berdasarkan Gambar 4.16, *endpoint* ini memvalidasi muatan data menggunakan skema *SaveLibraryRequest.* Setelah token pengguna diverifikasi, parameter uid diteruskan ke *modul save_to_library().* Proses ini dibungkus dalam blok *try-except* untuk menangkap potensi galat komunikasi dengan *server* basis data.

e. *Endpoint* Pengambilan Daftar Pustaka *(GET /library)*

Endpoint ini dirancang untuk mendistribusikan daftar lengkap senyawa yang telah dikurasi oleh pengguna, sebagaimana ditunjukkan pada Gambar 4.17 berikut.

```python
@app.get("/library")
def list_library(user=Depends(get_current_user)):
    uid = user["uid"]
    try:
        items = get_user_library(uid)
        return JSONResponse(content=jsonable_encoder(items))
    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Gagal mengambil data: {str(e)}")
```

Gambar 4. 17 Implementasi *Endpoint* Pengambilan Data Pustaka Pribadi

Berdasarkan Gambar 4.17, dengan menggunakan metode GET, fungsi list_library memanfaatkan uid dari sesi yang aktif untuk memanggil fungsi get_user_library(). Data dokumen (*dictionary*) yang dikembalikan dari *Firestore* kemudian diserialisasi menggunakan jsonable_encoder sebelum dikirimkan kembali ke *frontend* dalam format JSON.

f. *Endpoint* Penghapusan Pustaka Senyawa *(DELETE /library/{item_id})*

Manajemen data pustaka juga mencakup kemampuan untuk menghapus referensi senyawa yang sudah tidak relevan. Potongan kode *endpoint* ini ditunjukkan pada Gambar 4.18 berikut.

```python
@app.delete("/library/{item_id}")
def delete_from_library(item_id: str, user=Depends(get_current_user)):
    uid = user["uid"]
    try:
        remove_from_library(uid, item_id)
        return {"success": True}
    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Gagal menghapus: {str(e)}")
```

Gambar 4. 18 Implementasi *Endpoint* Penghapusan Data Pustaka

Berdasarkan Gambar 4.18, *endpoint* ini menggunakan metode DELETE dan memanfaatkan parameter jalur (*path parameter*) berupa {item_id}. Setelah

identitas pengguna dipastikan valid, sistem mengeksekusi penghapusan dokumen secara spesifik melalui modul remove_from_library().

g. *Endpoint* Pengecekan Status Pustaka *(GET /library/check)*

Untuk menyajikan antarmuka pengguna yang dinamis, *server* menyediakan rute khusus untuk memeriksa apakah sebuah senyawa *SMILES* sudah tersimpan di *Library* atau belum, sebagaimana ditunjukkan pada Gambar 4.19 berikut.

```python
@app.get("/library/check")
def check_saved_status(smiles: str, user=Depends(get_current_user)):
    uid = user["uid"]
    try:
        result = check_is_saved_by_smiles(uid, smiles)
        return result
    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Gagal cek status: {str(e)}")
```

Gambar 4. 19 Implementasi *Endpoint* Pengecekan Status Senyawa

Berdasarkan Gambar 4.19, fungsi check_saved_status menerima parameter kueri (*query parameter*) smiles. Fungsi ini memanggil check_is_saved_by_smiles() untuk melakukan verifikasi eksistensi dokumen pada *Firestore* tanpa harus memuat seluruh isi dokumen, sehingga komputasi berjalan sangat cepat.

h. *Endpoint* Riwayat Prediksi Pengguna *(GET /history/mine)*

Endpoint ini bertugas melayani antarmuka halaman riwayat pengguna secara personal dan terisolasi. Potongan kode *endpoint* ini ditunjukkan pada Gambar 4.20 berikut.

```python
@app.get("/history/mine")
def my_history(user=Depends(get_current_user)):
    uid = user["uid"]
    try:
        items = get_user_history(uid)
        return JSONResponse(content=jsonable_encoder(items))
    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Gagal mengambil riwayat: {str(e)}")
```

Gambar 4. 20 Implementasi *Endpoint* Pengambilan Riwayat Personal

Berdasarkan Gambar 4.20, fungsi my_history mengutamakan privasi pengguna dengan membatasi kueri pengambilan data (get_user_history()) hanya pada entri search_history yang memiliki kecocokan userId dengan token yang dikirim oleh klien.

i. *Endpoint* Manajemen Riwayat dan Pemeliharaan Skema *(GET /history)*

Sistem juga menyediakan *endpoint* riwayat sekunder yang mengimplementasikan mekanisme pertahanan skema data (*schema fallback*), sebagaimana ditunjukkan pada Gambar 4.21 berikut.

```python
@app.get("/history")
def history():
    data = get_history()

    # Backfill defaults biar history aman walau schema berubah
    for item in data:
        res = item.get("result", {})

        nc = res.get("new_compound", {})
        nc.setdefault("formula", "")
        nc.setdefault("molecular_weight", 0.0)
        nc.setdefault("tg", 0.0)
        nc.setdefault("tg_justification", "")
        nc.setdefault("pid", "")
        nc.setdefault("polymer_class", "")
        res["new_compound"] = nc

        scs = res.get("similar_compounds", [])
        for sc in scs:
            sc.setdefault("name", "")
            sc.setdefault("formula", "")
            sc.setdefault("molecular_weight", 0.0)
            sc.setdefault("tg", 0.0)
            sc.setdefault("pid", "")
            sc.setdefault("polymer_class", "")

            score = sc.get("similarity_score", 0.0)
            sc.setdefault("similarity_percent", float(score) * 100.0)

        res["similar_compounds"] = scs
        item["result"] = res

    return JSONResponse(content=jsonable_encoder(data))
```

Gambar 4. 21 Implementasi *Endpoint* Riwayat dengan Pertahanan Skema Data

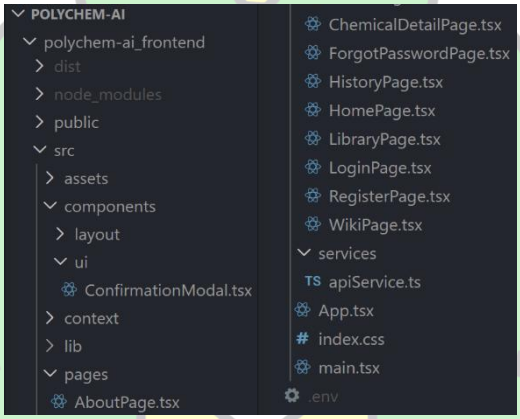
Pada Gambar 4.21, fungsi history() menerapkan logika *backfill defaults* menggunakan metode setdefault(). Fitur ini memastikan bahwa meskipun format *output* skema data dari kecerdasan buatan (*Gemini*) mengalami perubahan di masa depan atau terdapat field yang hilang, aplikasi klien tidak akan mengalami *crash* karena *server* secara proaktif menyisipkan nilai *default* yang aman (seperti *string* kosong "" atau angka 0.0).

4.1.3 Implementasi *Frontend*

Implementasi antarmuka pengguna (*frontend*) pada aplikasi web PolyChem dibangun menggunakan pustaka React berbasis TypeScript. Pengembangan antarmuka ini mengedepankan prinsip tata letak yang berpusat pada pengguna (*user-centered design*) guna memastikan peneliti dapat mengakses fitur kecerdasan buatan dengan efisien.

1. Struktur Arsitektur *Frontend*

Sistem menerapkan pendekatan arsitektur berbasis komponen (*component-based architecture*) yang memisahkan antara logika presentasi, pengelolaan status (*state management*), dan integrasi layanan API. Potongan struktur direktori proyek *frontend* disajikan pada Gambar 4.22 berikut.



Gambar 4. 22 Struktur Direktori Proyek *Frontend*

Berdasarkan Gambar 4.22, struktur direktori src dibagi menjadi beberapa modul utama untuk memudahkan pemeliharaan kode:

- components/: Berisi komponen UI yang dapat digunakan kembali (*reusable*), seperti DashboardLayout.tsx untuk tata letak kerangka utama aplikasi.
- pages/: Memuat komponen tingkat halaman (seperti HomePage.tsx, HistoryPage.tsx, dan LibraryPage.tsx) yang merepresentasikan setiap rute utama sistem.
- services/: Memuat file apiService.ts yang bertugas sebagai agen komunikasi eksklusif antara antarmuka klien dengan *backend* FastAPI. Seluruh interaksi dengan Firebase, baik untuk autentikasi maupun manipulasi data,

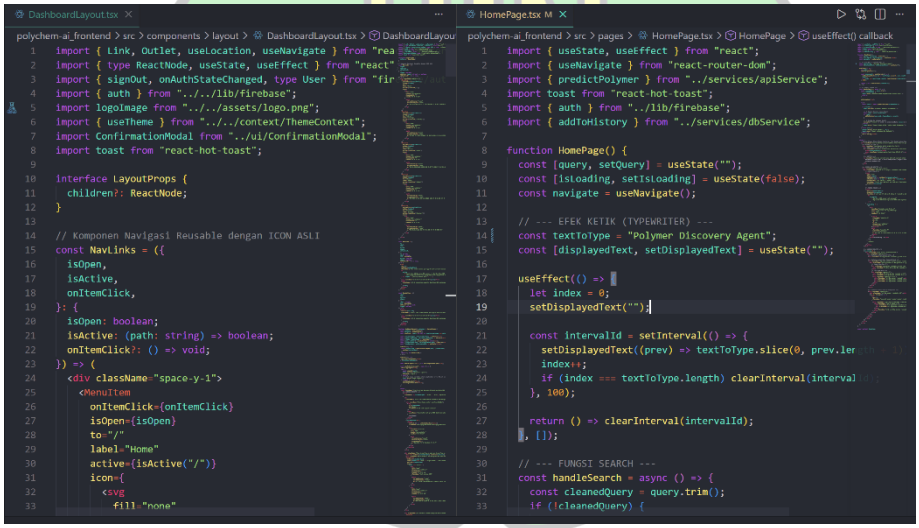
didelegasikan sepenuhnya kepada *backend* melalui berkas ini, sehingga *frontend* tidak lagi memiliki jalur komunikasi langsung ke Firestore.

2. Implementasi Halaman Utama (*Dashboard*)

Halaman utama merupakan realisasi dari rancangan *Home Screen* (Gambar 3.4) yang telah disusun pada tahap perancangan. Halaman ini bertindak sebagai pusat kendali di mana pengguna dapat bernavigasi dan langsung memulai proses prediksi senyawa.

a. Implementasi Komponen (Kode)

Pengembangan halaman ini melibatkan pembuatan komponen tata letak (*layout*) dan komponen halaman penelusuran. Potongan kode yang mengatur tata letak dan logika interaksi pada Halaman Utama disajikan pada Gambar 4.23 berikut.



Gambar 4. 23 Implementasi Kode Komponen *DashboardLayout* dan *HomePage*

b. Penjelasan Logika Kode

Berdasarkan Gambar 4.23, implementasi antarmuka dipisahkan ke dalam dua berkas utama untuk menjaga modularitas:

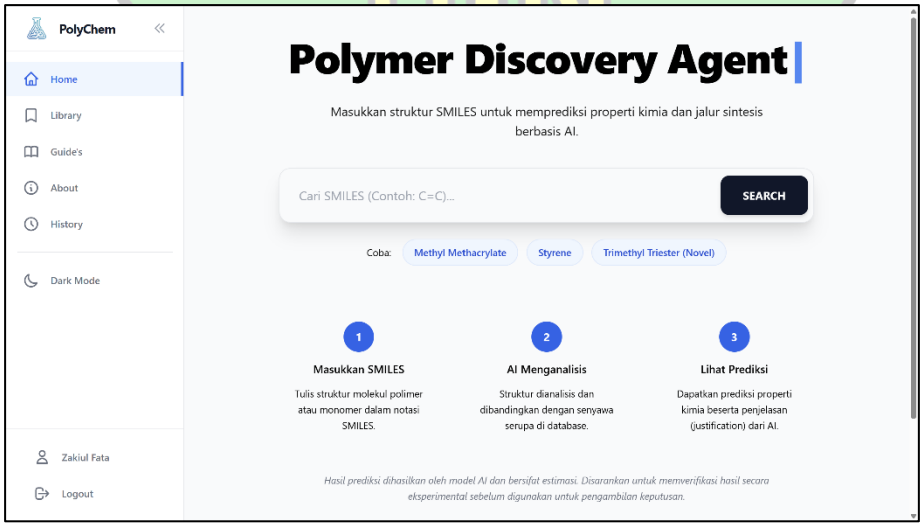
- Pada sisi kiri (DashboardLayout.tsx), sistem mendefinisikan kerangka aplikasi yang memuat navigasi *sidebar* persisten. Penggunaan komponen fungsional seperti NavLinks dan elemen rute memastikan pengguna dapat berpindah menu

(seperti *Home*, *Library*, *History*) secara dinamis tanpa perlu memuat ulang seluruh halaman (*Single Page Application*).

- Pada sisi kanan (HomePage.tsx), sistem mengelola antarmuka utama menggunakan *hooks* dari React. useState digunakan untuk menangkap *input* pengguna pada variabel query dan mengatur status pemuatan (isLoading). Selain itu, useEffect dimanfaatkan untuk memberikan interaksi visual berupa animasi pengetikan (*typewriter effect*) pada teks "Polymer Discovery Agent". Pada komponen ini pula, sistem mengimpor modul predictPolymer dan addToHistory dari layanan eksternal (services), yang nantinya akan dieksekusi oleh fungsi asinkron handleSearch.

c. Tampilan Antarmuka (*Output*)

Hasil kompilasi dari implementasi struktur dan logika kode tersebut menghasilkan antarmuka pengguna interaktif yang disajikan pada Gambar 4.24 berikut.



Gambar 4. 24 Antarmuka Halaman Utama (*Dashboard*)

Pada Gambar 4.24, DashboardLayout.tsx merender *sidebar* navigasi di sisi kiri (Home, Library, Guide's, About, History), sementara HomePage.tsx mengisi area kerja utama dengan judul aplikasi beranimasi *typewriter* dan kolom input *SMILES* yang diposisikan di tengah layar. Tersedia pula contoh cepat (*quick example*) berupa senyawa Methyl Methacrylate, Styrene, dan Trimethyl Triester agar pengguna dapat langsung mencoba fitur prediksi tanpa harus menulis notasi SMILES sendiri. Tiga langkah penggunaan (*Masukkan SMILES → AI*

Menganalisis → *Lihat Prediksi*) ditampilkan sebagai panduan singkat, dan disertai *disclaimer* bahwa hasil prediksi bersifat estimasi AI yang perlu diverifikasi secara eksperimental. Desain ini merepresentasikan pendekatan *user-centered design* yang menyederhanakan alur kerja peneliti sejak kontak pertama dengan aplikasi.

3. Implementasi Halaman Hasil Prediksi (*Output Screen*)

Halaman hasil prediksi merupakan antarmuka krusial yang bertugas menyajikan data analisis kimia dan justifikasi dari *Agentic AI*. Halaman ini merupakan realisasi langsung dari rancangan *Output Screen – Wireframe* pada Gambar 3.5.

a. Implementasi Komponen (Kode)

Pengembangan halaman ini bergantung pada penerimaan data JSON yang kompleks dari *backend* untuk diurai menjadi elemen visual yang terstruktur. Potongan kode yang menangani penerimaan dan pemrosesan data pada komponen ChemicalDetailPage.tsx disajikan pada Gambar 4.25 berikut.

polychem-ai_frontend > src > pages > ChemicalDetailPage.tsx > ChemicalDetailPage > handleSave

```
1   import { useLocation, useNavigate } from "react-router-dom";
2   import { useEffect, useState } from "react";
3   import toast from "react-hot-toast";
4   import { saveToLibrary } from "../services/dbService";
5   import { auth } from "../lib/firebase";
6   import { buildAssetUrl, normalizePrediction } from "../services/apiService";
7
8   function ChemicalDetailPage() {
9     const location = useLocation();
10    const navigate = useNavigate();
11    const [isSaving, setIsSaving] = useState(false);
12
13    const predictionData = normalizePrediction(location.state?.predictionData);
14
15    useEffect(() => {
16      if (!predictionData) {
17        navigate("/");
18      }
19    }, [predictionData, navigate]);
20
21    if (!predictionData) return null;
22
23    const compound = predictionData.new_compound;
24    const similar = predictionData.similar_compounds;
25
26    const handleSave = async () => {
```

Gambar 4. 25 Implementasi Kode Komponen *ChemicalDetailPage*

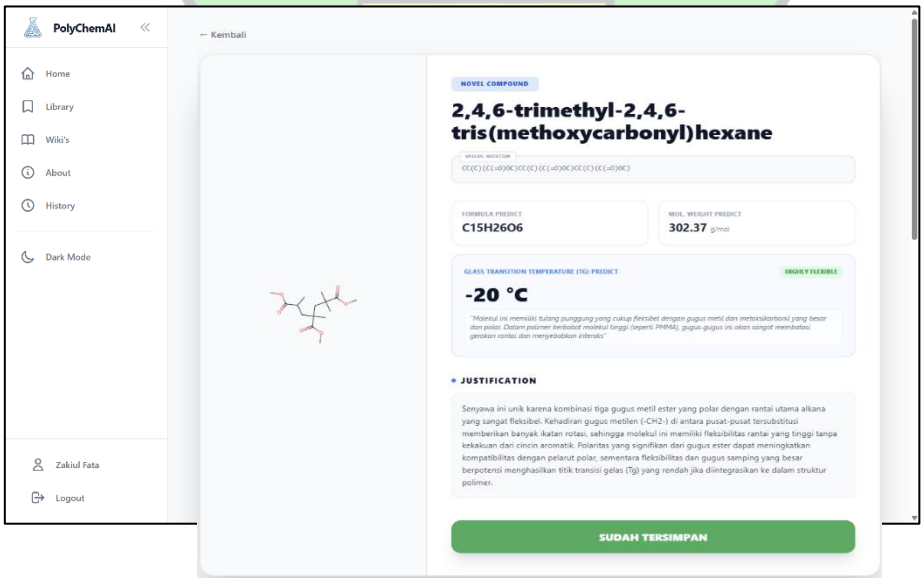
b. Penjelasan Logika Kode

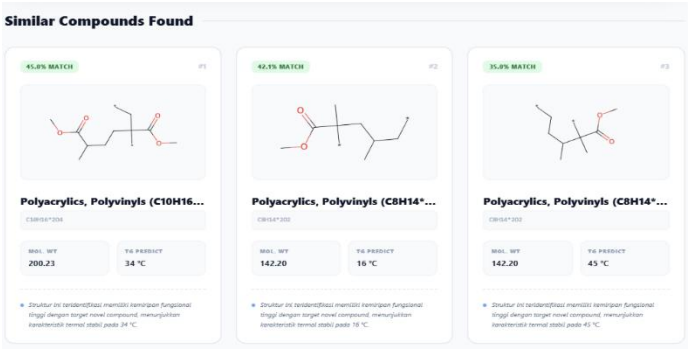
Berdasarkan Gambar 4.25, halaman ini menggunakan *hook* useLocation dari react-router-dom untuk menangkap objek data *state* yang dikirimkan dari halaman utama setelah proses *fetching* API selesai. Pendekatan ini sangat efisien karena sistem tidak perlu melakukan pemanggilan API berulang kali. Data yang diterima kemudian diproses melalui fungsi utilitas normalizePrediction() untuk memastikan format struktur objek sesuai dengan standar tipe data di *frontend*.

Sistem juga mengimplementasikan mekanisme perlindungan antarmuka (*UI safeguard*) menggunakan useEffect; jika variabel predictionData kosong (misalnya karena pengguna me-*refresh* halaman secara paksa), aplikasi akan secara otomatis mengarahkan (*redirect*) pengguna kembali ke halaman utama (navigate("/")). Selanjutnya, data dipecah menjadi dua variabel utama, yaitu compound untuk senyawa utama dan similar untuk menampung *array* senyawa dengan struktur yang mirip. Terdapat pula deklarasi handleSave yang nantinya akan memanggil fungsi saveToLibrary menuju basis data Firebase.

c. Tampilan Antarmuka (*Output*)

Hasil penguraian data (*parsing*) dari *backend* tersebut dirender menjadi antarmuka visual yang komprehensif, sebagaimana disajikan pada Gambar 4.26 berikut.





Gambar 4. 26 Antarmuka Halaman Hasil Prediksi Senyawa

Halaman ini merupakan luaran inti dari aplikasi web PolyChem, yang menerjemahkan data mentah hasil komputasi model AI menjadi informasi kimia yang terstruktur dan mudah diinterpretasikan oleh pengguna, baik yang memiliki latar belakang kimia maupun tidak. Secara umum, hasil prediksi terbagi ke dalam dua kartu utama dan satu bagian pembanding, sebagaimana disajikan pada Gambar 4.26:

a. Kartu Identitas Senyawa (Kiri)

Kartu ini menampilkan hasil identifikasi struktural dari senyawa yang diprediksi:

- Visualisasi Struktur 2D: Menggambarkan susunan atom dan ikatan kimia senyawa secara visual, sehingga peneliti dapat langsung mengenali bentuk molekul tanpa perlu menerjemahkan notasi SMILES secara manual.
- Label *"NOVEL COMPOUND"*: Menandakan bahwa senyawa hasil input belum tercatat dalam basis data pembanding sistem, sehingga hasil prediksi merupakan estimasi murni dari model AI, bukan pencocokan data yang sudah ada.
- Nama Sistematis & Notasi *SMILES*: Nama IUPAC senyawa serta representasi tekstualnya dalam notasi *Simplified Molecular Input Line Entry System* (SMILES), yaitu format standar untuk merepresentasikan struktur molekul sebagai deretan karakter.
- *Formula & Molecular Weight:* Menunjukkan komposisi atom penyusun senyawa (formula kimia) dan massa molekul relatifnya (g/mol), yang menjadi dasar untuk memperkirakan sifat fisik dan reaktivitas senyawa.

- *Glass Transition Temperature (Tg) Predict:* Menampilkan suhu transisi gelas hasil prediksi, yaitu suhu di mana material polimer berubah dari fase keras dan getas (*glassy*) menjadi lebih lunak dan fleksibel (*rubbery*). Nilai ini merupakan salah satu parameter paling penting dalam karakterisasi material polimer karena menentukan rentang suhu aplikasi material tersebut. Nilai Tg ini disertai indikator kategori fleksibilitas (misalnya *HIGHLY FLEXIBLE*) sebagai interpretasi cepat: semakin rendah nilai Tg, semakin fleksibel material pada suhu ruang, dan sebaliknya.

b. Kartu Justifikasi AI (Kanan)

Kartu ini menampilkan paragraf *Justification*, yaitu penjelasan naratif hasil penalaran model *Large Language Model* (Google Gemini 2.5 Flash) yang menguraikan alasan ilmiah di balik nilai properti yang diprediksi misalnya bagaimana gugus fungsi tertentu pada struktur molekul memengaruhi fleksibilitas rantai polimer. Bagian ini penting karena mengubah prediksi AI dari sekadar angka menjadi penjelasan yang dapat diverifikasi secara konseptual oleh peneliti. Tombol "Save to Library" memungkinkan pengguna menyimpan hasil analisis ini ke dalam koleksi pribadi untuk ditinjau kembali di kemudian hari.

c. Bagian Senyawa Serupa (Similar Compounds Found)

Di bagian bawah halaman, sistem menampilkan senyawa-senyawa pembanding yang paling mirip dengan hasil prediksi, dalam bentuk *card grid*. Setiap kartu memuat:

- Persentase Match: Nilai kemiripan struktural yang dihitung menggunakan metode *Tanimoto Similarity*, yaitu metrik standar dalam *cheminformatics* untuk mengukur kesamaan antar-struktur molekul berdasarkan *fingerprint* kimianya. Semakin tinggi persentase, semakin mirip struktur kedua senyawa.
- Struktur Molekul Pembanding, Berat Molekul, dan Prediksi Tg: Memberikan konteks kuantitatif agar peneliti dapat membandingkan sifat senyawa baru dengan senyawa yang sudah dikenal.
- Justifikasi Singkat: Penjelasan ringkas mengenai kemiripan fungsional antara senyawa yang diprediksi dan senyawa pembanding.

4. Implementasi Halaman Koleksi Senyawa (*Library*)

Halaman *Library* berfungsi sebagai repositori pribadi di mana pengguna dapat menyimpan, meninjau kembali, dan mengelola senyawa polimer yang telah diprediksi. Halaman ini merupakan bentuk realisasi langsung dari rancangan *Library Screen – Wireframe* pada Gambar 3.6 yang telah disusun pada tahap perancangan.

a. Implementasi Komponen (Kode)

Pengembangan komponen LibraryPage.tsx berfokus pada mekanisme pengambilan data (*data fetching*) secara asinkron dari basis data Firebase dan pengelolaan *state* antarmuka yang dinamis. Potongan kode inisialisasi dan deklarasi *state* disajikan pada Gambar 4.27 berikut.

polychem-ai_frontend > src > pages > LibraryPage.tsx > LibraryPage

```tsx
import { useEffect, useState } from "react";
import { Link, useNavigate } from "react-router-dom";
import { auth } from "../lib/firebase";
import { onAuthStateChanged } from "firebase/auth";
import {
  getUserLibrary,
  removeFromLibrary,
  type SavedChemical,
} from "../services/dbService";
import { buildAssetUrl } from "../services/apiService";

function LibraryPage() {
  const [items, setItems] = useState<SavedChemical[]>([]);
  const [loading, setLoading] = useState(true);
  const [userUid, setUserUid] = useState<string | null>(null);
  const navigate = useNavigate();

  // --- STATE FOR DELETE MODAL ---
  const [isModalOpen, setIsModalOpen] = useState(false);
  const [itemToDelete, setItemToDelete] = useState<string | null>(null);
  const [isDeleting, setIsDeleting] = useState(false);

  // --- 1. Fetch Data Function ---
  const fetchItems = async (uid: string) => {
```

Gambar 4. 27 Deklarasi Modul dan *State* pada Komponen LibraryPage

b. Penjelasan Logika Kode

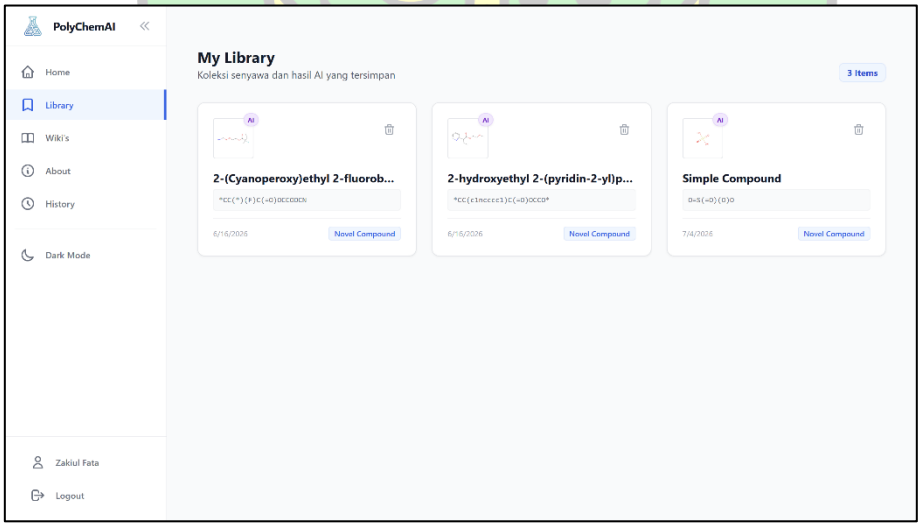
Berdasarkan Gambar 4.27 dan arsitektur kode sumber secara keseluruhan, halaman ini memanfaatkan *hook* useEffect yang terintegrasi dengan fungsi onAuthStateChanged dari Firebase. Hal ini memungkinkan sistem untuk secara *real-time* mendeteksi sesi pengguna yang sedang aktif. Sesuai dengan arsitektur *Backend-Mediated*, alih-alih melakukan kueri langsung ke *Firestore*, antarmuka kini memanggil layanan getUserLibraryBackend. Layanan ini akan mengeksekusi

permintaan HTTP GET menuju *endpoint* /library pada *server* FastAPI dengan menyertakan *ID Token* pengguna di dalam *header* otorisasi (Authorization: Bearer).

Selama proses pengambilan data dari *server* berlangsung, sistem memanfaatkan *state* loading untuk menampilkan animasi kerangka tata letak (*skeleton loader*). Implementasi *skeleton* berbasis *grid* ini diterapkan untuk mempertahankan struktur visual halaman saat memuat data, sehingga menghindari terjadinya *layout shift* yang mengganggu kenyamanan visual. Selain itu, komponen ini juga mendefinisikan logika perlindungan data melalui *state* isModalOpen dan itemToDelete, di mana setiap permintaan penghapusan (*delete*) dari pengguna akan ditahan sementara oleh *modal* konfirmasi sebelum dieksekusi secara permanen oleh *server* melalui fungsi removeFromLibraryBackend.

c. Tampilan Antarmuka (*Output*)

Setelah kueri basis data berhasil diselesaikan, *state* items diperbarui dan sistem merender antarmuka daftar senyawa secara komprehensif, sebagaimana disajikan pada Gambar 4.28 berikut.



Gambar 4. 28 Antarmuka Halaman *Library* Pengguna

Pada Gambar 4.28, koleksi senyawa milik peneliti ditampilkan menggunakan pendekatan tata letak *card grid*. Setiap *card* dirancang dengan visual hierarkis yang rapi, menyajikan gambar struktur molekul 2D, nama senyawa, notasi *SMILES*, dan tanggal penyimpanan (*timestamp*). Untuk memberikan konteks informasi yang cepat, setiap *card* dilengkapi dengan lencana (*badge*) kategorisasi seperti "AI" (jika senyawa dihasilkan oleh model kecerdasan buatan) dan "Novel

Compound". Di sudut kanan atas setiap *card*, terdapat tombol hapus ikon tempat sampah yang terhubung dengan sistem *modal* konfirmasi, memastikan pengguna memiliki interaksi yang aman dan terkendali terhadap basis data koleksi pribadi mereka.

5. Implementasi Halaman Riwayat Riset (*History Screen*)

Halaman Riwayat berfungsi sebagai log aktivitas otomatis yang merekam seluruh jejak prediksi yang pernah dilakukan oleh sistem *Agentic AI*. Implementasi halaman ini merupakan bentuk nyata dari rancangan *History Screen – Wireframe* pada Gambar 3.7 yang telah disusun sebelumnya.

a. Implementasi Komponen (Kode)

Pengembangan halaman riwayat berfokus pada pengambilan data sekumpulan riwayat pengguna dan mekanisme navigasi kembali ke halaman hasil prediksi. Potongan kode inisialisasi *state* dan logika klik pada HistoryPage.tsx disajikan pada Gambar 4.29 berikut.

Gambar 4. 29 Implementasi Kode Komponen HistoryPage

b. Penjelasan Logika Kode

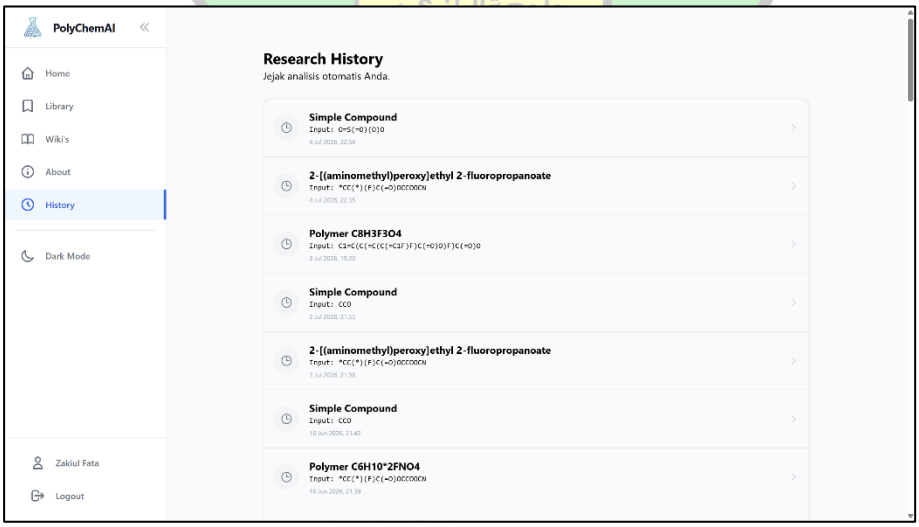
Berdasarkan Gambar 4.29, halaman ini menerapkan *hook* useEffect untuk memantau status autentikasi melalui fungsi onAuthStateChanged. Jika sesi

pengguna valid, sistem memanggil fungsi getUserHistoryBackend yang akan mengirimkan permintaan HTTP GET menuju *endpoint* /history/mine di *server* FastAPI untuk mengunduh daftar riwayat yang diamankan. Jika sesi tidak valid, pengguna akan langsung dialihkan secara paksa ke halaman otorisasi (navigate("/login")), untuk memastikan keamanan akses data.

Salah satu implementasi logika yang paling esensial pada komponen ini terletak pada fungsi handleItemClick. Ketika pengguna memilih salah satu riwayat pada daftar, sistem tidak melakukan pemanggilan ulang (*re-fetch*) ke *backend* atau API kecerdasan buatan. Sebaliknya, antarmuka mengambil *string* JSON mentah dari objek item.full_data, mengurainya secara lokal menggunakan JSON.parse(), lalu menormalisasinya dengan fungsi normalizePrediction(). Data yang sudah terstruktur ini kemudian dikirimkan secara langsung ke rute hasil (/result) melalui manajemen *state routing*. Pendekatan ini secara drastis menghemat laju *bandwidth* dan mempercepat waktu muat halaman (*load time*), memberikan pengalaman pengguna yang sangat responsif.

c. Tampilan Antarmuka (*Output*)

Setelah data berhasil diambil dan diproses, sistem menampilkannya dalam bentuk daftar (*list view*) yang rapi, sebagaimana disajikan pada Gambar 4.30 berikut.



Gambar 4. 30 Antarmuka Halaman Riwayat Riset (*History Screen*)

Pada Gambar 4.30, data riwayat disajikan secara vertikal untuk memudahkan pengguna dalam membaca riwayat secara kronologis. Setiap baris daftar menampilkan informasi kunci yang paling dibutuhkan oleh peneliti: nama senyawa (*result name*), input *SMILES* awal yang digunakan, serta waktu analisis (*timestamp*). Halaman ini juga dilengkapi dengan interaksi visual berupa efek *hover*, di mana saat kursor diarahkan ke salah satu baris, akan muncul teks "Lihat Hasil" beserta ikon panah yang memberikan isyarat visual (*affordance*) yang jelas bahwa elemen tersebut dapat diklik untuk melihat detail analisis secara penuh.

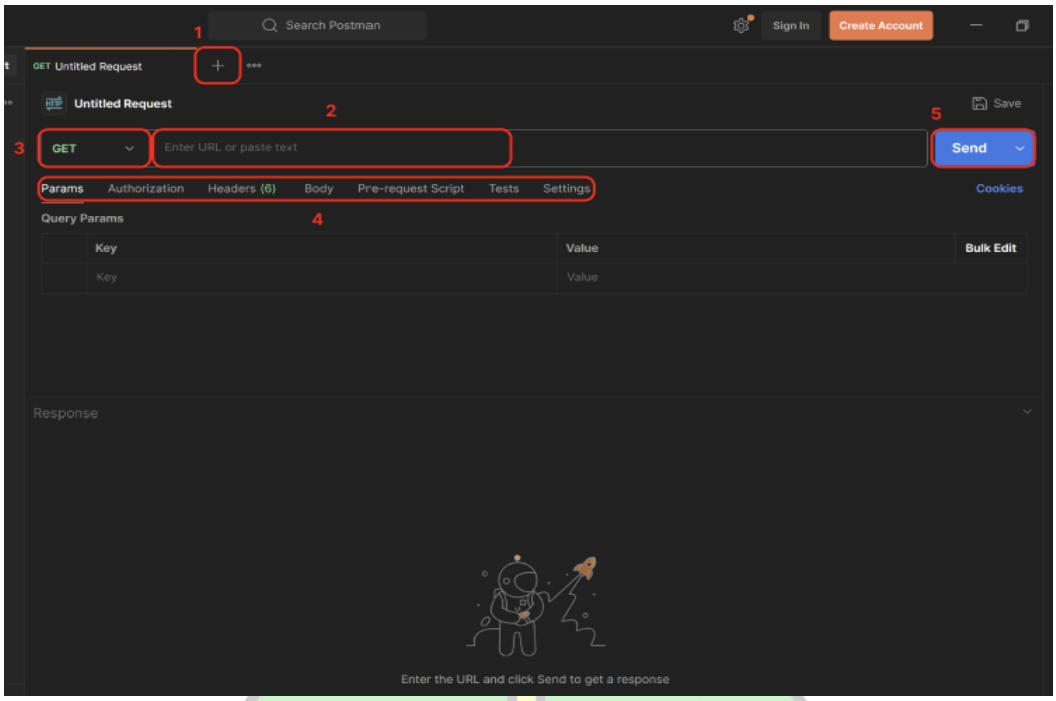
4.2 Pengujian Sistem

Tahap pengujian sistem dilakukan untuk memverifikasi dan memvalidasi bahwa aplikasi PolyChem yang telah diimplementasikan dapat berjalan sesuai dengan spesifikasi kebutuhan fungsional yang telah didefinisikan pada tahap perancangan (Bab III). Metode pengujian yang digunakan dalam penelitian ini adalah *Black Box Testing*, yaitu pengujian yang berfokus pada evaluasi input dan output sistem tanpa perlu meninjau struktur logika kode internalnya. Pengujian ini dibagi menjadi dua aspek utama, yaitu pengujian pada sisi *backend* (API) dan pengujian pada sisi *frontend* (Web).

4.2.1 Pengujian Fungsional API (*Backend*)

Pengujian fungsionalitas antarmuka pemrograman aplikasi (API) pada sisi *backend* dilakukan menggunakan perangkat lunak Postman. Postman bertindak sebagai klien *REST API* yang memungkinkan simulasi pengiriman *HTTP Request* ke *endpoint server* FastAPI secara langsung. Pengujian ini menggunakan metode *Black Box Testing*, di mana evaluasi difokuskan pada kesesuaian respons sistem terhadap variasi masukan pengguna tanpa meninjau struktur logika internal kode.

Sebelum melaksanakan skenario pengujian, diperlukan pemahaman mengenai lingkungan pengujian pada aplikasi Postman. Anatomi antarmuka yang digunakan sebagai instrumen pengujian disajikan pada Gambar 4.31 berikut.



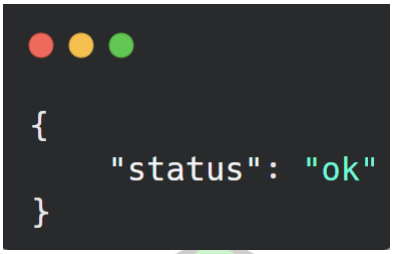
Gambar 4. 31 Antarmuka Lingkungan Pengujian API pada Postman

- Tab Pengujian Baru (1): Tombol yang digunakan untuk membuka *tab* ruang kerja baru untuk setiap *endpoint* API yang akan diverifikasi secara independen.
- Kolom URL (2): Area untuk memasukkan alamat *endpoint server* lokal yang sedang diuji, seperti http://localhost:8000/predict untuk fungsi prediksi.
- Metode HTTP (3): Menu *dropdown* untuk menentukan jenis protokol komunikasi yang digunakan. Pada sistem ini, pengujian mengutamakan metode POST untuk proses inferensi data (pengiriman SMILES) dan metode GET untuk pengambilan data riwayat maupun status sistem.
- Panel Konfigurasi *Request* (4): Area untuk mengatur parameter *request* yang mencakup *Authorization* (untuk memasukkan *Bearer Token*), *Headers*, dan *Body* (untuk menyisipkan data masukan simulasi dalam format JSON pada metode POST).
- Tombol Send (5): Tombol pemicu (*eksekutor*) untuk mengirimkan seluruh parameter *request* menuju *backend*. Hasil dari eksekusi ini akan memunculkan respons pada area bawah (*Response*), yang memuat kode status HTTP dan luaran JSON dari sistem *Agentic AI.*

1. Pengujian Ketersediaan Layanan *(GET /Health)*

Gambar 4.32 memperlihatkan respons yang di berikan oleh server saat melakukan proses pengecekan status kesiapan layanan (*health check*). *Endpoint*

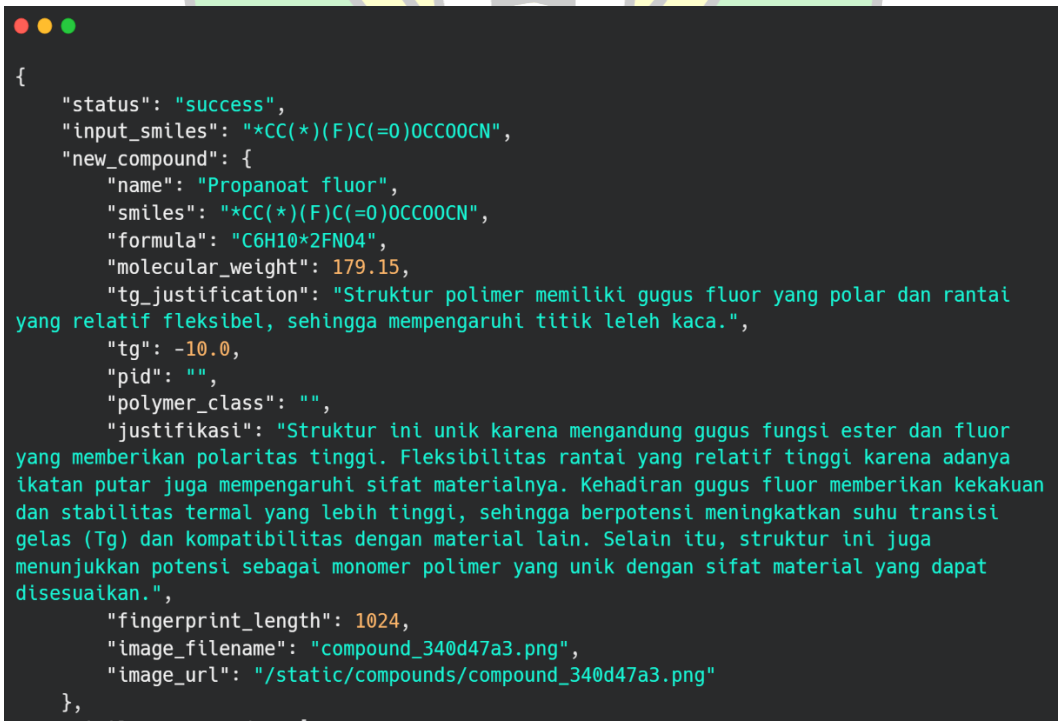
API ini dapat diakses melalui *Uniform Resource Locator* (URL) dengan format http://{domain}/health. Untuk mengirim permintaan ke *endpoint* tersebut kita dapat metode HTTP GET. Saat permintaan berhasil, server mengembalikan respons berformat JSON yang berisi sebuah atribut *status* bertipe string yang memuat informasi status operasional server saat ini.



Gambar 4. 32 Hasil Pengujian Endpoint */health*

2. Pengujian *Endpoint* Prediksi Senyawa *(POST /predict)*

Gambar 4.33 memperlihatkan respons yang di berikan oleh server saat melakukan proses prediksi senyawa baru (*novel compound*). *Endpoint* API ini dapat diakses melalui URL dengan format http://{domain}/predict. Untuk mengirim permintaan ke *endpoint* tersebut, digunakan metode HTTP POST.



Gambar 4. 33 Hasil Pengujian Endpoint */predict* bagian *novel compound*

Respons yang diterima berupa data dalam format JSON sebagaimana ditunjukkan pada Gambar 4.33. Struktur data ini terdiri atas atribut *status* dan *input_smiles*, serta sebuah objek utama bernama *new_compound* yang mendeskripsikan hasil prediksi. Objek *new_compound* ini direpresentasikan oleh beberapa atribut utama, yaitu: (1) *name*, bertipe string yang memuat nama senyawa; (2) *smiles*, bertipe string sebagai representasi struktur kimia; (3) *formula*, bertipe string yang memuat rumus molekul; (4) *molecular_weight*, bertipe *float* yang menunjukkan berat molekul; (5) *tg*, bertipe *float* sebagai nilai prediksi suhu transisi gelas; dan (6) *justifikasi*, bertipe string yang menjelaskan makna atau narasi saintifik dari hasil prediksi tersebut. Selain itu, terdapat juga atribut *image_url*, bertipe string yang memuat url sebagai gambar pendukung visual.

Gambar 4.34 dibawah memperlihatkan lanjutan respons yang di berikan oleh server dari hasil prediksi yang sama, berfokus pada pencarian senyawa serupa (*similar compounds*).


```
    "similar_compounds": [
        {
            "rank": 1,
            "smiles": "*CC(*)(F)C(=O)OCCC",
            "name": "Other polymers (C6H9*2FO2)",
            "formula": "C6H9*2FO2",
            "molecular_weight": 132.13,
            "tg": 62.0,
            "pid": "P522021",
            "polymer_class": "Other polymers",
            "similarity_score": 0.6060606060606061,
            "similarity_percent": 60.60606060606061,
            "justifikasi": "Senyawa ini adalah ester (motif ester) yang memiliki gugus
propanoat terfluorinasi (motif fluorinasi dan pola rantai propanoat). Rantai asamnya adalah
2-fluoropropanoat, dan gugus alkoholnya adalah propil.",
            "image_filename": "compound_c24b0112.png",
            "image_url": "/static/compounds/compound_c24b0112.png"
        },
        {
            "rank": 2,
            "smiles": "*CC(*)(F)C(=O)OCC(Cl)(Cl)Cl",
            "name": "Other polymers (C5H4*2Cl3FO2)",
            "formula": "C5H4*2Cl3FO2",
            "molecular_weight": 221.44,
            "tg": 127.0,
            "pid": "P522055",
            "polymer_class": "Other polymers",
            "similarity_score": 0.5588235294117647,
            "similarity_percent": 55.88235294117647,
            "justifikasi": "Senyawa ini merupakan ester (motif ester) dengan bagian asam 2-
fluoropropanoat (motif fluorinasi dan pola rantai propanoat). Kehadiran banyak atom halogen
(F dan Cl) pada molekul ini juga meningkatkan polaritasnya (motif polaritas).",
            "image_filename": "compound_d645f155.png",
            "image_url": "/static/compounds/compound_d645f155.png"
        },
        {
```

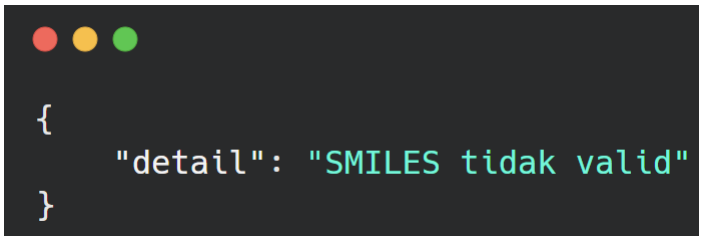
```
        {
            "rank": 3,
            "smiles": "*CC(*)(F)C(=O)OCC",
            "name": "Other polymers (C5H7*2FO2)",
            "formula": "C5H7*2FO2",
            "molecular_weight": 118.11,
            "tg": 94.0,
            "pid": "P522020",
            "polymer_class": "Other polymers",
            "similarity_score": 0.5454545454545454,
            "similarity_percent": 54.54545454545454,
            "justifikasi": "Senyawa ini adalah ester (motif ester) yang mengandung gugus
propanoat terfluorinasi (motif fluorinasi dan pola rantai propanoat). Bagian asamnya adalah
2-fluoropropanoat, dan bagian alkoholnya adalah etil.",
            "image_filename": "compound_59342a72.png",
            "image_url": "/static/compounds/compound_59342a72.png"
        }
    ]
}
```


Gambar 4. 34 Hasil Pengujian Endpoint */predict* bagian *similar compounds*

Respons yang diterima berupa data dalam format JSON dengan atribut *similar_compounds*, yang bertipe *array of object*. Setiap objek di dalam *array similar_compounds* mewakili data pembanding (peringkat 1, 2, dan 3) yang memiliki atribut utama, yaitu: (a) *rank*, bertipe *integer* sebagai penanda unik urutan peringkat; (b) *name* dan *smiles*, bertipe string yang memuat nama serta notasi struktur; (c) *tg*, bertipe *float* yang merepresentasikan nilai suhu transisi gelas; (d) *similarity_percent*, bertipe *float* yang menunjukkan persentase tingkat kecocokan struktur; dan (e) *justifikasi*, yang berisi representasi alasan kemiripan struktur senyawa tersebut dengan target prediksi. Setiap objek di dalam *array* ini juga dilengkapi dengan *image_url*, bertipe string yang memuat url gambar item senyawa pembanding.

3. Pengujian Penanganan Kesalahan (*Error Handling*) pada Input *SMILES*

Gambar 4.35 memperlihatkan respons yang diberikan oleh server saat melakukan proses penanganan kesalahan (*error handling*) pada fitur prediksi. Pada skenario pengujian ini, validasi sistem diuji dengan memasukkan format struktur molekul (SMILES) yang tidak valid, yaitu C1CCCCC. *Endpoint* API ini dapat diakses melalui URL dengan format http://{domain}/predict menggunakan metode HTTP POST.

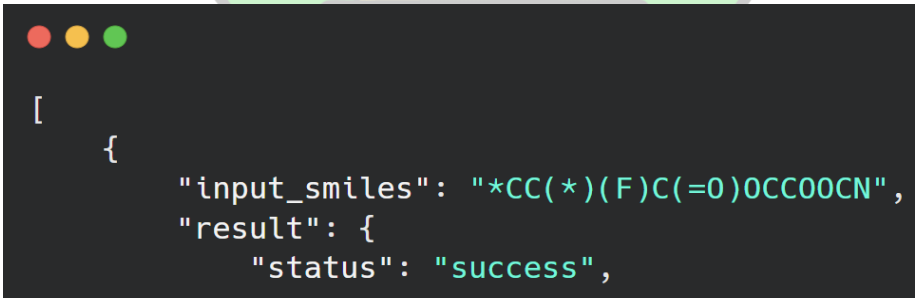


Gambar 4. 35 Hasil Pengujian Penanganan Kesalahan Input *SMILES*

Saat sistem mendeteksi adanya kegagalan validasi masukan tersebut di sisi *backend*, proses komputasi akan langsung dihentikan secara otomatis agar tidak membebani agen AI. *server* kemudian mengembalikan respons berformat JSON yang berisi rincian penolakan. Struktur data ini sangat sederhana karena hanya direpresentasikan oleh satu atribut utama, yaitu *detail*, yang bertipe string dan memuat deskripsi spesifik mengenai penyebab kegagalan berupa pesan "SMILES tidak valid". Atribut ini berfungsi penting untuk memberikan umpan balik (*feedback*) yang jelas kepada sistem *frontend* sehingga dapat menampilkan peringatan yang sesuai kepada pengguna.

4. Pengujian *Endpoint* Riwayat Riset *(GET /history)*

Gambar 4.36 memperlihatkan respons yang diberikan oleh server saat melakukan proses pengambilan data log aktivitas prediksi melalui *endpoint* API /history. *Endpoint* ini dapat diakses melalui URL dengan format http://{domain}/history menggunakan metode HTTP GET.



Gambar 4. 36 Hasil Pengujian *Endpoint /history*

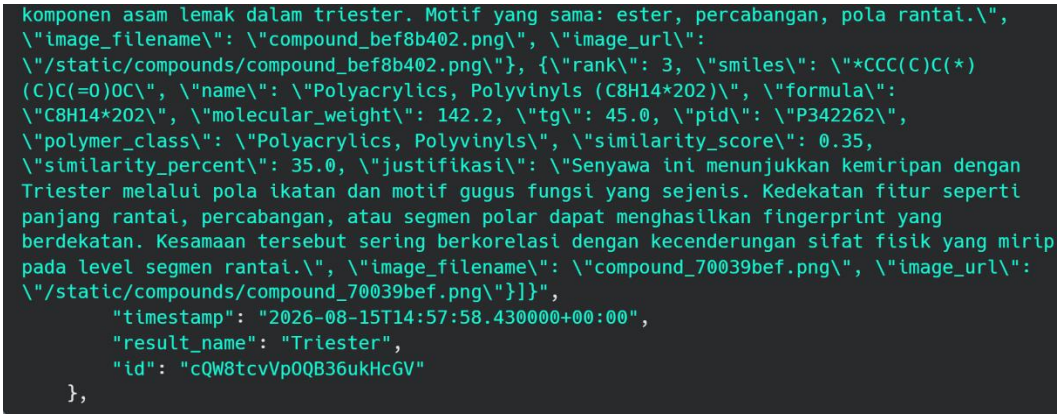
Berbeda dengan fitur riwayat personal, *endpoint* ini beroperasi pada level sistem global tanpa memfilter identitas pengguna dan tidak memerlukan autentikasi. Fungsi utama *endpoint* ini terbagi menjadi dua: pertama, sebagai mekanisme *cache-check* untuk mengoptimalkan kinerja server dan menghemat

pemakaian kuota API. Jika struktur SMILES yang sama diproses ulang dalam kurun waktu 10 menit terakhir, sistem akan mengambil hasil dari *cache* alih-alih menjalankan ulang *pipeline* komputasi (RDKit, Tanimoto, dan Gemini). Kedua, *endpoint* ini berfungsi sebagai log aktivitas yang menampilkan daftar *request* prediksi terbaru yang diproses oleh server secara keseluruhan. Penyimpanan data *cache* ini tidak menggunakan pangkalan data utama, melainkan menggunakan memori lokal (*diskcache*) yang membatasi kapasitas maksimal sebanyak tiga entri terbaru dengan masa simpan (*Time-To-Live*/TTL) selama 10 menit.

5. Pengujian *Endpoint* Riwayat Prediksi Personal *(GET /history/mine)*

Gambar 4.37 memperlihatkan respons yang diberikan oleh server saat melakukan proses pengambilan data riwayat prediksi yang bersifat personal untuk setiap pengguna. *Endpoint* API ini dapat diakses melalui URL dengan format http://{domain}/history/mine menggunakan metode HTTP GET. Untuk memastikan bahwa data yang diambil terisolasi dan hanya milik pengguna yang bersangkutan (*tenant isolation*), permintaan ini mewajibkan penyertaan token autentikasi yang sah pada *header* permintaan.


```
"userId": "7pbOFBEXVTQ0bFf8qob9rDhPKDw1",
        "result_smiles": "CC(C)(C(=O)OC)CC(C)(C(=O)OC)CC(C)(C(=O)OC)",
        "query": "CC(C)(C(=O)OC)CC(C)(C(=O)OC)CC(C)(C(=O)OC)",
        "full_data": "{\"status\": \"success\", \"input_smiles\": \"CC(C)(C(=O)OC)CC(C)
(C(=O)OC)CC(C)(C(=O)OC)\", \"new_compound\": {\"name\": \"Triester\", \"smiles\": \"CC(C)
(C(=O)OC)CC(C)(C(=O)OC)CC(C)(C(=O)OC)\", \"formula\": \"C15H26O6\", \"molecular_weight\":
302.37, \"tg_justification\": \"Struktur triester memiliki rantai yang relatif fleksibel dan
gugus ester yang polar, namun tidak memiliki cincin yang kaku. Hal ini memungkinkan molekul
untuk bergerak dengan lebih bebas, sehingga menurunkan titik gelasnya.\", \"tg\": -20.0,
\"pid\": \"\", \"polymer_class\": \"\", \"justifikasi\": \"Struktur ini unik karena memiliki
tiga gugus ester yang polar, yang dapat mempengaruhi sifat material dan kompatibilitasnya.
Fleksibilitas rantai yang relatif tinggi karena tidak adanya cincin atau ikatan aromatik
juga memungkinkan molekul ini memiliki tingkat kebebasan gerak yang lebih tinggi. Hal ini
dapat berimplikasi pada penurunan suhu transisi gelas (Tg) dan meningkatkan stabilitas
termal. Selain itu, kehadiran gugus ester juga dapat mempengaruhi sifat hidrofilik dan
lipofilik molekul, sehingga memungkinkan aplikasi dalam berbagai bidang seperti polimer dan
bahan kimia.\", \"fingerprint_length\": 1024, \"image_filename\": \"compound_3be1c416.png\",
\"image_url\": \"/static/compounds/compound_3be1c416.png\"}, \"similar_compounds\":
[{\"rank\": 1, \"smiles\": \"*CC(*)(CCC(C)C(=O)OC)C(=O)OC\", \"name\": \"Polyacrylics,
Polyvinyls (C10H16*2O4)\", \"formula\": \"C10H16*2O4\", \"molecular_weight\": 200.23,
\"tg\": 34.0, \"pid\": \"P044506\", \"polymer_class\": \"Polyacrylics, Polyvinyls\",
\"similarity_score\": 0.45, \"similarity_percent\": 45.0, \"justifikasi\": \"Molekul ini
memiliki dua gugus ester (`C(=O)OC`), yang menunjukkan kemiripan dengan target
\\\"Triester\\\" yang mengandung banyak gugus ester. Selain itu, terdapat percabangan
signifikan pada rantai karbonnya, pola yang umum ditemukan pada triester kompleks. Motif
yang sama: ester, percabangan, pola rantai.\", \"image_filename\":
\"compound_fe5638e3.png\", \"image_url\": \"/static/compounds/compound_fe5638e3.png\"},
{\"rank\": 2, \"smiles\": \"*CC(C)CC(*)(C)C(=O)OC\", \"name\": \"Polyacrylics, Polyvinyls
(C8H14*2O2)\", \"formula\": \"C8H14*2O2\", \"molecular_weight\": 142.2, \"tg\": 16.0,
\"pid\": \"P342268\", \"polymer_class\": \"Polyacrylics, Polyvinyls\", \"similarity_score\":
0.42105263157894735, \"similarity_percent\": 42.10526315789473, \"justifikasi\": \"Molekul
ini mengandung satu gugus ester (`C(=O)OC`), yang merupakan unit fungsional utama dari
triester. Rantai karbonnya juga menunjukkan percabangan, karakteristik yang sering ada pada
```



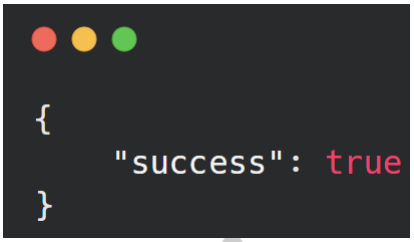
Gambar 4. 37 Hasil Pengujian *Endpoint /history/mine*

Respons yang diterima berupa data dalam format JSON dengan struktur *array of object* sebagaimana ditunjukkan pada Gambar 4.37. Setiap objek di dalam *array* tersebut merepresentasikan satu rekaman riwayat riset dan memiliki beberapa atribut utama, yaitu: userId bertipe string sebagai penanda unik kepemilikan data pengguna, result_smiles bertipe string yang memuat notasi struktur molekul hasil prediksi, query bertipe string yang memuat struktur SMILES awal yang diinputkan pengguna saat pencarian, full_data bertipe string yang memuat data komputasi prediksi secara utuh (*stringified JSON*), timestamp bertipe string yang menunjukkan waktu spesifik kapan rekaman aktivitas tersebut disimpan ke basis data, result_name bertipe string yang memuat nama senyawa hasil prediksi, dan id bertipe string sebagai penanda identitas unik dokumen riwayat pada sistem.

Penggunaan tipe data string untuk atribut full_data (*stringified JSON*) merupakan pendekatan yang sangat efisien dalam pengiriman *payload* data yang kompleks. Nantinya, data mentah ini akan diurai (*parse*) kembali secara lokal di sisi klien. Pendekatan ini memungkinkan komponen antarmuka (*User Interface*) PolyChem untuk merender tata letak halaman detail riwayat secara dinamis dan responsif, tanpa perlu membebani *server* dengan pemanggilan *endpoint* tambahan secara berulang.

6. Pengujian *Endpoint* Pustaka Senyawa *(POST /library)*

Gambar 4.38 memperlihatkan respons yang diberikan oleh server saat melakukan proses penyimpanan senyawa hasil prediksi ke dalam koleksi pustaka pribadi pengguna (*library*). *Endpoint* API ini dapat diakses melalui URL dengan format http://{domain}/library menggunakan metode HTTP POST.



Gambar 4. 38 Hasil Pengujian *Endpoint POST /library*

Untuk mengeksekusi proses penyimpanan ini, sistem menerapkan protokol keamanan yang mewajibkan penyertaan token autentikasi (*Bearer Token*) pada *header* permintaan. Selain itu, klien juga harus mengirimkan *payload* data berformat JSON pada bagian *body*, yang pada skenario pengujian ini memuat atribut "smiles" dengan nilai "*CC(*)(F)C(=O)OCCOOCN". Hal ini bertujuan untuk memastikan bahwa senyawa yang disimpan benar-benar tervalidasi dan diotorisasi sesuai dengan identitas sesi pengguna yang sedang aktif.

Saat permintaan berhasil diproses dan data senyawa tersimpan dengan aman ke dalam basis data, *server* mengembalikan respons berformat JSON sebagaimana ditunjukkan pada Gambar 4.38. Struktur data respons ini dibuat sangat ringkas dan efisien, hanya direpresentasikan oleh satu atribut utama yaitu success, yang bertipe *boolean* dengan nilai true. Atribut ini berfungsi sebagai indikator atau umpan balik (*feedback*) positif bagi antarmuka *frontend* bahwa operasi penambahan data ke pustaka telah berhasil dieksekusi tanpa adanya kendala operasional.

7. Pengujian *Endpoint* Pustaka Senyawa *(GET /library)*

Gambar 4.39 memperlihatkan respons yang diberikan oleh server saat melakukan proses pengambilan daftar koleksi senyawa (*library*) yang telah disimpan sebelumnya oleh pengguna. *Endpoint* API ini dapat diakses melalui URL dengan format http://{domain}/library menggunakan metode HTTP GET.

Gambar 4. 39 Hasil Pengujian *Endpoint GET /library*

Sama halnya dengan proses penyimpanan, *endpoint* ini menerapkan protokol keamanan yang ketat. Permintaan sistem mewajibkan penyertaan token autentikasi (*Bearer Token*) pada *header*. Pendekatan ini memastikan berjalannya isolasi data (*tenant isolation*), di mana *server* hanya akan mengambil dan mengembalikan dokumen pustaka yang memiliki kecocokan dengan identitas sesi pengguna yang sedang aktif tersebut.

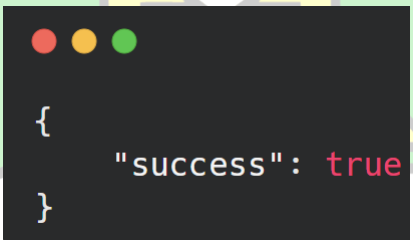
Saat permintaan berhasil dieksekusi, *server* mengembalikan respons berupa data dalam format JSON dengan struktur *array of object*, sebagaimana ditunjukkan pada gambar. Setiap objek di dalam *array* ini merepresentasikan satu entri senyawa yang tersimpan dan memiliki beberapa atribut utama, yaitu: userId bertipe string sebagai penanda unik identitas pengguna pemilik data, savedAt bertipe string yang menunjukkan stempel waktu (*timestamp*) spesifik kapan senyawa tersebut disimpan, score bertipe *null* (atau *float*) yang disiapkan untuk menampung nilai metrik evaluasi senyawa, id bertipe string sebagai identitas unik (*primary key*) dokumen di dalam basis data, category bertipe *null* (atau string) untuk pengelompokan jenis senyawa, properties bertipe *null* (atau *object*) yang dapat memuat detail sifat fisikokimia tambahan, isAiResult bertipe *boolean* (dalam hal

ini bernilai false) yang bertindak sebagai indikator apakah senyawa tersebut merupakan hasil dari generasi kecerdasan buatan atau kurasi eksternal, image dan image_url bertipe string kosong (atau *path* URL) yang disiapkan untuk memuat rujukan visual struktur molekul 2D, smiles bertipe string yang memuat notasi struktur molekul (*contoh:* *CC(*)(F)C(=O)OCCOOCN), dan name bertipe *null* (atau string) yang disiapkan untuk menampung nama sistematis senyawa.

Struktur pengembalian data yang memperbolehkan nilai *null* atau *string* kosong ini membuktikan bahwa *server* dirancang dengan pertahanan skema data yang baik, sehingga aplikasi antarmuka (*frontend*) tidak akan mengalami galat (crash) meskipun terdapat beberapa informasi detail senyawa yang belum lengkap.

8. Pengujian *Endpoint* Penghapusan Pustaka *(DELETE /library/{item_id})*

Gambar 4.40 memperlihatkan respons yang diberikan oleh server saat melakukan proses penghapusan data senyawa polimer dari koleksi pustaka pribadi pengguna. *Endpoint* API ini dapat diakses melalui URL dengan format http://{domain}/library/{item_id} menggunakan metode HTTP DELETE. Pada rancangan *endpoint* ini, parameter {item_id} bertindak sebagai parameter jalur (*path parameter*) yang dinamis untuk menunjuk secara spesifik identitas unik dokumen senyawa yang ingin dihapus dari basis data.



Gambar 4. 40 Hasil Pengujian *Endpoint DELETE /library*

Untuk mengeksekusi operasi penghapusan ini secara aman, sistem kembali menerapkan mekanisme otorisasi yang ketat dengan mewajibkan penyertaan token autentikasi (*Bearer Token*) pada *header* permintaan. Protokol keamanan ini sangat krusial untuk memvalidasi hak akses (*access control*), guna memastikan bahwa pengguna hanya diizinkan untuk menghapus data senyawa yang memang benar-benar berada di bawah kepemilikan sesi mereka sendiri.

Saat proses verifikasi berhasil dan dokumen senyawa telah dihapus secara permanen dari basis data, *server* mengembalikan respons dalam format JSON sebagaimana ditunjukkan pada Gambar 4.40. Struktur respons ini dirancang sangat ringkas, hanya memuat satu atribut utama yaitu success yang bertipe *boolean* dengan nilai true. Nilai kembalian ini berfungsi sebagai indikator atau umpan balik positif bagi sistem antarmuka (*frontend*) pada PolyChem bahwa operasi penghapusan telah sukses dieksekusi, sehingga antarmuka dapat langsung memperbarui (*re-render*) tampilan daftar koleksi tanpa perlu memuat ulang seluruh halaman.

9. Pengujian *Endpoint* Pengecekan Status *(GET /library/check)*

Gambar 4.41 memperlihatkan respons yang diberikan oleh server saat mengeksekusi fitur pengecekan status untuk memastikan apakah suatu senyawa polimer telah tersimpan di dalam pustaka pribadi pengguna atau belum. *Endpoint* API ini dapat diakses melalui URL dengan format http://{domain}/library/check menggunakan metode HTTP GET.

Gambar 4. 41 Hasil Pengujian *Endpoint library/check*

Untuk melakukan pengecekan ini secara aman, sistem mewajibkan penyertaan token autentikasi (*Bearer Token*) pada *header* permintaan. Selanjutnya, *endpoint* ini memerlukan parameter kueri (*query parameter*) dengan kunci smiles yang berisi notasi struktur molekul yang ingin diperiksa. Pada pengujian

fungsionalitas ini, dilakukan dua skenario simulasi: skenario pertama dengan memasukkan nilai *CC(*)(F)C(=O)OCCOOCN untuk memvalidasi kondisi saat molekul sudah pernah disimpan, dan skenario kedua dengan memasukkan nilai CCCC untuk memvalidasi kondisi saat molekul belum ada di dalam koleksi.

Saat permintaan berhasil dieksekusi, *server* mengembalikan respons dalam format JSON yang memuat dua blok keluaran sesuai dengan hasil pengecekan. Struktur data respons ini direpresentasikan oleh dua atribut utama, yaitu: isSaved bertipe *boolean* yang bertindak sebagai indikator utama. Atribut ini bernilai true jika senyawa berhasil ditemukan dalam basis data pengguna, dan bernilai false jika senyawa tersebut belum tersimpan, serta itemId bertipe string yang memuat identitas unik dokumen (seperti "ai_340d47a3b4144483") yang dihasilkan secara otomatis oleh sistem berdasarkan struktur molekul tersebut.

Logika pengembalian status *boolean* yang dikombinasikan dengan *itemId* ini sangat krusial bagi antarmuka *frontend* PolyChem. Pendekatan ini memungkinkan aplikasi untuk memberikan umpan balik visual secara instan kepada pengguna (misalnya, mengubah status tombol dari *Save to Library* menjadi *Saved*) tanpa perlu membebani *server* dengan melakukan kueri pengambilan keseluruhan daftar isi pustaka.

10. Pengujian *Endpoint* Sinkronisasi Profil (POST /auth/sync-profile)

Gambar 4.42 memperlihatkan respons yang diberikan oleh server saat mengeksekusi fitur sinkronisasi data identitas pengguna yang diperoleh melalui penyedia otorisasi pihak ketiga (Google SSO) ke dalam basis data Firestore. *Endpoint* API ini dapat diakses melalui URL dengan format http://{domain}/auth/sync-profile menggunakan metode HTTP POST. Skenario pengujian ini dilakukan dengan mengirimkan data profil melalui *request body* dan mewajibkan penyertaan token autentikasi (*Bearer Token* atau ID Token) pada *header* permintaan. Pendekatan ini memastikan bahwa proses pembaruan data dilakukan melalui jalur otorisasi yang sah.

Gambar 4. 42 Hasil Pengujian *Endpoint* Sinkronisasi Profil Pengguna

Saat permintaan berhasil diproses, *server* mengembalikan respons dengan status keberhasilan dalam format JSON. Struktur data respons ini sangat jelas, direpresentasikan oleh atribut success bertipe *boolean* yang bernilai true, serta sebuah objek profile yang memuat struktur data identitas pengguna secara lengkap.

Objek profile tersebut direpresentasikan oleh enam atribut utama yang keseluruhannya bertipe string, yaitu:uid, sebagai penanda identitas unik pengguna pada sistem; name, yang memuat nama lengkap pengguna; photoURL, yang memuat tautan (*URL*) foto profil pengguna dari akun Google; createdAt, yang menunjukkan stempel waktu (*timestamp*) spesifik kapan data profil tersebut dicatat atau disinkronkan ke dalam basis data;role, yang menentukan tingkat hak akses atau peran pengguna di dalam aplikasi (dalam hal ini bernilai "user"); dan email, yang memuat alamat surel pengguna yang terdaftar. Hasil keluaran JSON ini secara langsung memvalidasi bahwa setiap sesi pengguna yang baru masuk (*login*) akan berhasil dipetakan dan disinkronkan ke dalam basis data sistem untuk mendukung personalisasi fitur pada aplikasi PolyChem.

11. Rekapitulasi Hasil Pengujian API

Secara keseluruhan, pengujian *Black Box* pada sisi *backend* menunjukkan bahwa fungsionalitas antarmuka pemrograman aplikasi (API) telah berjalan sesuai dengan spesifikasi kebutuhan yang dirancang. *server* terbukti tangguh dalam menangani skenario eksekusi normal maupun saat menerima anomali masukan. Seluruh hasil pengujian pada berbagai *endpoint* utama aplikasi web PolyChem dirangkum pada Tabel 4.2 berikut.

Tabel 4. 2 Rekapitulasi Hasil Pengujian Fungsional API

No	*Endpoint* & Metode	Skenario Pengujian	Hasil yang Diharapkan	Hasil Aktual (Postman)	Status
1	GET /health	Memeriksa ketersediaan dan kesiapan layanan *server* FastAPI.	Mengembalikan HTTP Status 200 OK dan struktur JSON {"status": "ok"}.	HTTP 200 OK, pesan status sesuai.	*Valid*
2	POST /predict	Mengirim data JSON dengan format *SMILES* valid (*CC(*)(F)C(=O)OCCOOCN).	Mengembalikan HTTP Status 200 OK beserta data properti polimer dan justifikasi dari agen AI.	HTTP 200 OK, struktur JSON respons lengkap dan valid.	*Valid*
3	POST /predict	Mengirim data JSON dengan format *SMILES* tidak valid (C1CCCCC).	Mengembalikan HTTP Status 400 Bad Request dengan deskripsi penolakan untuk mencegah *crash*.	HTTP 400 Bad Request, muncul pesan "SMILES tidak valid".	*Valid*

4	GET /history	Mengambil data riwayat dengan parameter identitas pengguna (*User ID*) yang sah.	Mengembalikan HTTP Status 200 OK berisi *array* objek riwayat pencarian pengguna.	HTTP 200 OK, *array* JSON riwayat tampil dengan lengkap.	*Valid*
5	GET /history /mine	Mengambil data riwayat kueri prediksi personal berdasarkan sesi token yang aktif.	Mengembalikan HTTP Status 200 OK berisi rekaman log AI khusus milik pengguna tersebut.	HTTP 200 OK, riwayat riset personal berhasil dimuat dengan aman.	*Valid*
6	POST /library	Mengirim data kurasi senyawa kimia polimer baru untuk disimpan ke pustaka personal.	Mengembalikan HTTP Status 200 OK dan menyimpan dokumen ke koleksi dengan parameter pengenal terisolasi.	HTTP 200 OK, pesan respons mengembalikan status sukses.	*Valid*

7	GET /library	Mengambil daftar pustaka senyawa pribadi berdasarkan sesi token pengguna yang aktif.	Mengembalikan HTTP Status 200 OK berisi daftar lengkap koleksi senyawa milik pengguna.	HTTP 200 OK, data terambil secara terisolasi sesuai hak akses pengguna.	*Valid*
8	DELETE /library /{item_id}	Mengirimkan permintaan penghapusan entri senyawa tertentu melalui parameter jalur (*path parameter*).	Mengembalikan HTTP Status 200 OK dan menghapus dokumen senyawa dari basis data.	HTTP 200 OK, data berhasil dihapus dari koleksi pustaka.	*Valid*
9	GET /library/ check	Memeriksa status kurasi molekul berdasarkan parameter kueri *SMILES*.	Mengembalikan HTTP Status 200 OK dan nilai status *boolean* kesamaan data (true/false).	HTTP 200 OK, status kepemilikan data terdeteksi secara akurat.	*Valid*
10	POST /auth /sync-profile	Mengirimkan data identitas dari Google SSO disertai *ID Token* otorisasi yang sah.	Mengembalikan HTTP Status 200 OK dan menyinkron	HTTP 200 OK, profil pengguna berhasil dipetakan	*Valid*

			kan profil pengguna ke koleksi basis data.	ke *Firestore*.	
11	POST /predict (Uji Keandalan *Multi-Provider*)	Memaksa gangguan koneksi pada server utama (simulasi *timeout* atau *limit* pada agen Gemini).	Sistem memotong waktu tunggu (*fail-fast*) dan mengambil alih tugas pemrosesan melalui agen AI cadangan (Groq) tanpa memicu *crash*.	HTTP 200 OK, respons JSON lengkap berisi nama, Tg, dan justifikasi berhasil dihasilkan oleh Groq.	*Valid*

Berdasarkan Tabel 4.2, dapat disimpulkan bahwa arsitektur *backend* sistem telah siap dan andal untuk dikonsumsi oleh antarmuka web (*frontend*), dengan tingkat keberhasilan penanganan interupsi (*exception handling*) yang sangat baik saat mengorkestrasi komputasi kimia dan kecerdasan buatan.

4.2.2 Pengujian Fungsional Sistem

Pengujian fungsionalitas sistem dilakukan untuk memvalidasi interaksi aplikasi secara keseluruhan dari sudut pandang pengguna akhir (*end-user*). Karena representasi visual dari elemen antarmuka telah dipaparkan secara mendetail pada tahap implementasi, pengujian pada tahap ini difokuskan murni pada *Black Box Testing* dengan pendekatan *End-to-End* (E2E).

Pengujian ini bertujuan untuk memastikan bahwa seluruh alur kerja operasional aplikasi mulai dari proses autentikasi pengguna, validasi formulir

masukan, eksekusi pemanggilan *server* (*API Call*), hingga manajemen *state* aplikasi (seperti riwayat dan pustaka senyawa) dapat berjalan dengan mulus secara terintegrasi tanpa adanya galat atau kegagalan sistem (*crash*). Secara keseluruhan, hasil pengujian fungsionalitas aplikasi web PolyChem dirangkum pada Tabel 4.3 berikut.

Tabel 4. 3 Rekapitulasi Hasil Pengujian Fungsional Sistem

No	Modul / Fitur	Skenario Pengujian	Hasil yang Diharapkan	Hasil Aktual	Status
1	Autentikasi (Registrasi)	Memasukkan data pengguna baru (Nama, Email, *Password*) dan menekan *Register*.	Sistem mengirimkan kredensial ke Firebase, membuat profil di *Firestore*, lalu menampilkan notifikasi sukses.	Akun tercipta, notifikasi sukses muncul, dan diarahkan ke *Login*.	*Valid*
2	Autentikasi (*Login*)	Memasukkan *Email* dan *Password* yang valid.	Sistem memvalidasi sesi melalui Firebase Auth dan mengarahkan pengguna ke Halaman *Dashboard*.	Sesi tervalidasi, halaman berpindah ke *Dashboard*.	*Valid*

3	Prediksi (Validasi *Input*)	Menekan tombol *Search* pada Halaman Utama dengan kondisi kolom masukan *SMILES* kosong.	Sistem melakukan intersepsi (*client-side validation*) dengan menampilkan peringatan *Toast Error* warna merah tanpa mengirim *request* ke *server*.	Peringatan *Toast* muncul, sistem tidak *crash*.	*Valid*
4	Prediksi (Eksekusi AI)	Memasukkan struktur *SMILES* yang valid (cth: *CC(*)(F)C(=O)OC COOCN) lalu menekan *Search*.	Aplikasi menampilkan animasi *loading*, memanggil *API Backend*, lalu merender Halaman Hasil Prediksi secara utuh (Sifat *Tg*, Justifikasi, *Similar Compounds*).	Transisi ke Halaman Hasil Prediksi sukses, seluruh data terender sempurna.	*Valid*
5	Validasi Hak Akses	Menekan tombol *Save to Library* dalam keadaan	Sistem memblokir penyimpanan ke *database*	Aksi diblokir, *Toast Error*	

	(*Access Control*)	belum *login* (pengguna anonim).	dan memunculkan *Toast Error* "Silakan login untuk menyimpan."	penolakan akses muncul.	*Valid*
6	*Library* (Penyimpanan)	Menekan tombol *Save to Library* pada Halaman Hasil Prediksi senyawa (keadaan sudah *login*).	Sistem memicu fungsi *database* asinkron untuk menyimpan ke *Firestore* dan tombol memberikan umpan balik visual (*Toast* sukses).	Senyawa berhasil tersimpan dan muncul di daftar *Library*.	*Valid*
7	*Library* (Penghapusan)	Menekan ikon hapus (tempat sampah) pada salah satu kartu senyawa di *Library*.	Sistem memunculkan *Modal* konfirmasi. Jika disetujui, data akan dihapus secara permanen dari basis data dan antarmuka diperbarui.	*Modal* berfungsi, senyawa terhapus dengan mulus.	*Valid*
8	*History* (Riwayat)	Membuka tab *History* dari menu	Sistem mengambil	Riwayat masa lalu	

		navigasi samping (*Sidebar*).	data riwayat otomatis dari *server* dan merendernya dalam bentuk daftar riwayat interaktif.	tampil secara berurutan (*descending*).	*Valid*
9	Proteksi Data (*UI Safeguard*)	Melakukan *refresh* peramban secara paksa (*F5*) saat berada di tengah Halaman Hasil Prediksi.	Sistem mendeteksi bahwa *state* data hilang dan secara otomatis mengembalikan (*redirect*) pengguna ke *Dashboard* utama.	Pengguna dikembalikan ke Halaman Utama dengan aman.	*Valid*
10	Logout	Menekan tombol *Logout* pada menu bilah navigasi.	Sistem membersihkan seluruh sesi (*cache* & token), lalu mengarahkan paksa pengguna ke Halaman *Login*.	Sesi terputus total, diarahkan ke Halaman *Login*.	*Valid*

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian sistem yang telah dilakukan pada aplikasi PolyChem, dapat ditarik beberapa kesimpulan utama sebagai berikut:

1. Penelitian ini telah berhasil merancang dan membangun aplikasi web PolyChem menggunakan arsitektur *Client-Server* yang terpisah (*decoupled architecture*). Antarmuka pengguna (*frontend*) dibangun secara interaktif menggunakan React.js dan Tailwind CSS, sementara logika bisnis dan pemrosesan data (*backend*) diorkestrasi menggunakan *framework* FastAPI yang terbukti efisien.
2. Integrasi Agentic AI dengan pendekatan *neuro-symbolic* berhasil diterapkan dengan tangguh (*resilient*). Sistem tidak hanya mengandalkan model bahasa generatif, tetapi memadukan RDKit untuk validasi struktural kimia (ekstraksi Morgan Fingerprint dan Tanimoto Similarity) dengan Google Gemini 2.5 Flash API serta Groq API sebagai agen penalaran yang saling menopang (*fail-safe*) untuk menghasilkan prediksi nilai Glass Transition Temperature (Tg) beserta justifikasi saintifiknya secara berkelanjutan tanpa interupsi.
3. Layanan *Cloud* Firebase terbukti andal dalam mendukung infrastruktur sistem. *Firebase Authentication* berhasil mengelola siklus hidup identitas pengguna secara aman, sementara basis data *Firestore* (NoSQL) memfasilitasi penyimpanan riwayat prediksi (*Search History*) secara otonom dan koleksi senyawa pribadi (*Saved Chemicals*) secara terstruktur.
4. Berdasarkan hasil evaluasi menggunakan *Black Box Testing* secara *End-to-End* (E2E), sistem terbukti tangguh (*robust*). Seluruh fungsionalitas utama berjalan valid. Mekanisme penanganan kesalahan (*error handling*) pada *backend* saat menerima masukan anomali, serta validasi hak akses (*access control*) pada *frontend* saat pengguna anonim mencoba menyimpan data, mampu melindungi aplikasi dari kegagalan sistem (*crash*) dan menjaga integritas data.

5. Secara keseluruhan, aplikasi PolyChem berhasil mengatasi batasan alat prediksi berbasis *Command Line Interface* (CLI) konvensional dengan menyajikan antarmuka visual yang intuitif bagi peneliti. Meskipun demikian, hasil dari sistem ini masih bersifat prediksi *in silico* awal dan belum dapat sepenuhnya menggantikan tahapan eksperimen laboratorium yang sesungguhnya.

5.2 Saran

Meskipun aplikasi PolyChem telah berfungsi dengan baik, masih terdapat beberapa keterbatasan. Berikut adalah saran yang dapat direkomendasikan untuk pengembangan sistem pada penelitian selanjutnya:

1. Peningkatan Skala Dataset dan Model: Penelitian selanjutnya disarankan menggunakan *dataset* polimer yang lebih masif dan beragam. Selain itu, transisi dari penggunaan LLM umum (Google Gemini) menuju model AI yang secara khusus dilatih (*fine-tuned*) pada domain material kimia sangat direkomendasikan untuk meningkatkan akurasi prediksi.
2. Ekspansi Parameter Prediksi: Sistem dapat dikembangkan lebih lanjut untuk memprediksi metrik properti fisikokimia material lainnya di luar nilai Tg, seperti *Melting Temperature* (Tm), konduktivitas termal, *tensile strength*, maupun kelarutan polimer.
3. Pengujian Non-Fungsional dan Evaluasi Ahli: Karena penelitian ini masih berfokus pada pengujian fungsionalitas teknis (*Black Box*), pengembangan berikutnya perlu melakukan Pengujian Penerimaan Pengguna (*User Acceptance Testing*/UAT) yang melibatkan pakar/ahli di bidang kimia material. Selain itu, pengujian beban (*Load/Stress Testing*) pada *server* FastAPI diperlukan sebelum sistem diimplementasikan pada lingkungan produksi berskala besar.
4. Validasi Eksperimental (*Wet Lab*): Guna mengukur tingkat akurasi (tingkat galat/ *error rate*) prediksi secara kuantitatif, hasil keluaran dari PolyChem harus divalidasi dan dibandingkan secara langsung dengan data empiris dari hasil eksperimen sintesis di laboratorium fisik.
5. Peningkatan Keamanan Berbasis Peran (*Role-Based Security*): Skema keamanan saat ini telah menerapkan *Firestore Security Rules* yang menolak

seluruh akses langsung dari klien, sehingga seluruh operasi data terpusat dan terverifikasi melalui *backend*. Penelitian selanjutnya disarankan mengembangkan lapisan otorisasi lanjutan berupa kendali akses berbasis peran (*role-based access control*) misalnya membedakan hak akses antara peran *peneliti* dan *administrator* serta menerapkan pembatasan laju permintaan (*Rate Limiting*) pada *endpoint* API untuk mencegah eksploitasi beban oleh pengguna anonim.

6. Evaluasi Kinerja Multi-Provider pada Lingkungan Produksi: Keterbatasan kuota (*free tier*) dari layanan API utama telah berhasil dimitigasi secara efektif melalui implementasi lapisan *fallback* menggunakan Groq API, sehingga sistem tetap stabil tanpa langsung memicu *fallback* heuristik RDKit. Untuk pengembangan ke skala *enterprise* yang lebih masif, penelitian selanjutnya disarankan melakukan pengujian beban (*load testing*) khusus pada mekanisme peralihan *multi-provider* ini, serta mempertimbangkan penggunaan LLM *open-source* lokal (*on-premise*) untuk menekan angka latensi serendah mungkin.

DAFTAR PUSTAKA

Bandi, A., Kongari, B., Naguru, R., Pasnoor, S., & Vilipala, S. V. (2025). *The Rise of Agentic AI : A Review of Definitions , Frameworks , Architectures , Applications , Evaluation Metrics , and Challenges*. 1–50.

Bran, A. M., Cox, S., Schilter, O., Baldassari, C., White, A. D., & Schwaller, P. (2024). Augmenting large language models with chemistry tools. *ACM Transactions on Software Engineering and Methodology*, *33*(May). https://doi.org/10.1038/s42256-024-00832-8

Buendía-García, F., & Piris-Ruano, J. (2025). *Using Generative AI to Support UX Design Students in Web Development Courses*. 1–22.

Google Developers. (2024). *Firebase Documentation*. https://firebase.google.com/docs

Hou, X., Zhao, Y., Liu, Y. U. E., Yang, Z., Wang, K., Li, L. I., Luo, X., Lo, D., Grundy, J., & Wang, H. (2024). Large Language Models for Software Engineering: A Systematic Literature Review. *ACM Transactions on Software Engineering and Methodology*, *X*(December), 1–79.

Hua, J. (2024). *Extra dataset with SMILES, Tg, PID, Polimers Class*. https://www.kaggle.com/datasets/jinghua/extra-dataset-with-smiles-tg-pid-polimers-class

Jablonka, K. M., Ai, Q., Al-feghali, A., Badhwar, S., Bocarsly, J. D., Bran, A. M., Bringuier, S., Brinson, L. C., Choudhary, K., Circi, D., Cox, S., Jong, W. A. De, Evans, M. L., Gastellu, N., Genzling, J., Gil, V., Hong, Z., Imran, A., Kruschwitz, S., … Warren, S. (2023). *Electronic Supplementary Material (ESI) for Digital Discovery . This journal is © The Royal Society of Chemistry 2023 Supporting Information for : 14 Examples of How LLMs Can Transform Materials Science and Chemistry : A Reflection on a Large Language Model Hackathon*. 1–46.

Landrum, G. (2024). *RDKit: Open-source cheminformatics*. https://www.rdkit.org

Maulana, B. A., Mawarni, E., Hidayattuloh, M. Y., & Suryawijaya, V. (2023). *Pengujian Black Box pada Sistem Informasi Barang Berbasis Web Menggunakan Metode Boundary Value Analysis*. *2*(6), 1747–1753.

Meta Open Source. (2024). *React: The Library for Web and Native User Interfaces*. https://react.dev

Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill Education.

Richards, M., & Ford, N. (2020). *Fundamentals of Software Architecture: An Engineering Approach*. O'Reilly Media.

Safitri, K. I., & Harkespan, A. (2024). *Pengembangan Web Service Menggunakan Framework FastAPI untuk Meningkatkan Kemudahan*. *12*(2), 149–157.

Sarker, I. H. (2022). AI - Based Modeling : Techniques , Applications and Research Issues Towards Automation , Intelligent and Smart Systems. *SN Computer Science*, *3*(2), 1–20. https://doi.org/10.1007/s42979-022-01043-x

Schneider, S. L., Pablo, J. De, Ai, M. L., Schneider, L., Walsh, D., Olsen, B., & Pablo, J. De. (2024). *As featured in : Digital Discovery Generative BigSMILES : an extension for polymer*. https://doi.org/10.1039/d3dd00147d

Suryotomo, A. P., Akbar, B. M., & Husaini, R. (2024). *Performance Analysis of FastAPI Framework on Lost Circulation Handling Management Application in Oil Well Drilling*. *21*(1), 110–121. https://doi.org/10.31515/telematika.v21i1.13259

Tailwind Labs. (2024). *Tailwind CSS: Rapidly Build Modern Websites without Ever Leaving your HTML*. https://tailwindcss.com

Wang, K., & Li, L. I. (2024). *Large Language Models for Software Engineering: A Systematic Literature Review*. *33*(8). https://doi.org/10.1145/3695988