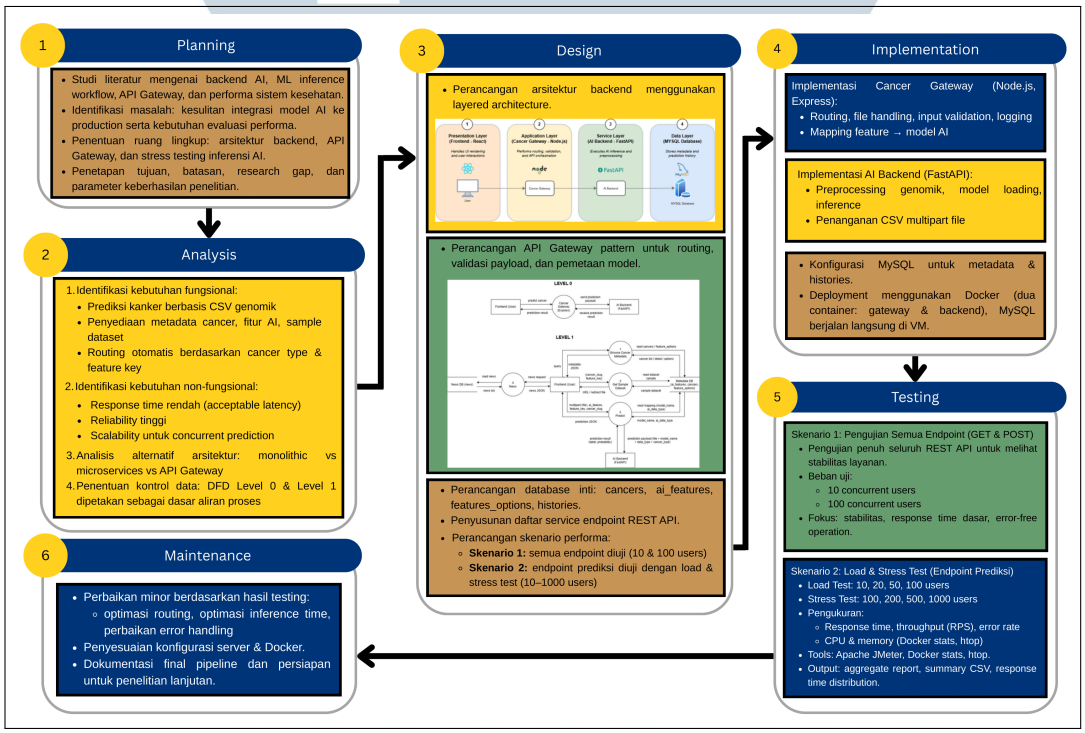
BAB 3
METODE PENELITIAN

3.1 Metode Penelitian

Penelitian ini menggunakan pendekatan *Software Development Life Cycle* (SDLC) sebagai dasar metodologis dalam pengembangan sistem AIRA. SDLC memberikan kerangka terstruktur mulai dari tahap perencanaan hingga evaluasi sistem, sehingga seluruh proses penelitian dapat dilakukan secara sistematis, terukur, dan dapat direplikasi. Model SDLC yang digunakan terdiri dari enam tahap utama yaitu *planning*, *analysis*, *design*, *implementation*, *testing*, dan *maintenance*. Alur penelitian divisualisasikan dalam bentuk *research pipeline* yang menggambarkan hubungan antar tahapan secara menyeluruh.



Gambar 3.1. Research Pipeline Penelitian

Pada Gambar 3.1 ditunjukkan alur penelitian yang digunakan dalam pengembangan dan evaluasi sistem AIRA. Research pipeline ini menggambarkan tahapan penelitian secara sistematis, mulai dari tahap perencanaan hingga evaluasi performa sistem. Setiap tahapan dirancang untuk memastikan bahwa pengujian performa arsitektur backend dilakukan secara terstruktur dan terukur.

1. *Planning*

   Tahap *planning* dilakukan secara iteratif dengan fokus pada identifikasi permasalahan utama, penentuan ruang lingkup awal, serta prioritas pengembangan sistem AIRA. Pada tahap ini dilakukan studi literatur terkait arsitektur *backend* untuk *machine learning inference*, pola *API gateway*, serta metode evaluasi performa sistem. Hasil dari tahap ini berupa tujuan penelitian, batasan penelitian, serta hipotesis awal yang dapat disesuaikan kembali pada iterasi berikutnya berdasarkan temuan selama pengembangan dan pengujian.

2. *Analysis*

   Tahap *analysis* bertujuan untuk mengidentifikasi kebutuhan sistem secara bertahap dan dapat diperbarui seiring berjalannya proses pengembangan. Kebutuhan fungsional mencakup kemampuan sistem dalam menangani berbagai jenis kanker, *routing* permintaan berdasarkan *cancer type* dan *feature key*, serta penyediaan layanan metadata dan prediksi. Kebutuhan non-fungsional difokuskan pada performa, *scalability*, dan keandalan sistem, yang dianalisis dengan mempertimbangkan berbagai alternatif arsitektur serta *trade-offs* yang mungkin muncul selama iterasi pengembangan.

3. *Design*

   Tahap *design* dilakukan secara inkremental dengan merancang dan menyempurnakan arsitektur *backend* berdasarkan hasil analisis dan umpan balik dari iterasi sebelumnya. Perancangan mencakup *layered architecture*, *API gateway*, komponen AI Backend, serta skema basis data. Aliran data antar komponen dimodelkan menggunakan DFD Level 0 dan Level 1 sebagaimana ditunjukkan pada Gambar 3.4 dan Gambar 3.5. Selain itu, pada tahap ini juga dirancang daftar layanan (*service endpoints*) dan skenario awal *load testing* serta *stress testing* yang dapat disesuaikan kembali pada iterasi selanjutnya.

4. *Implementation*

   Tahap *implementation* dilakukan melalui pengembangan modul sistem secara bertahap dan terpisah. *Cancer Gateway* dikembangkan menggunakan Node.js dan Express, sementara AI Backend dibangun menggunakan FastAPI dan model TensorFlow. Setiap komponen diimplementasikan, diuji secara

mandiri, dan kemudian diintegrasikan secara bertahap. Seluruh layanan dikemas menggunakan Docker untuk memastikan konsistensi lingkungan, dengan Docker Compose digunakan untuk mengelola orkestrasi multi-container pada satu server.

5. *Testing*

   Tahap *testing* dilakukan secara berulang pada setiap iterasi pengembangan untuk mengevaluasi stabilitas dan performa sistem. Pengujian mencakup seluruh *REST API* (GET dan POST) pada berbagai tingkat beban untuk mengukur *response time*, *throughput*, dan *error rate*. Selain itu, pengujian performa mendalam dilakukan pada *endpoint* prediksi sebagai inti sistem melalui skenario *load testing* dan *stress testing*. Skenario tambahan diterapkan untuk membandingkan beban komputasi antar *model* prediksi dengan konfigurasi *workload* yang sama, sehingga perbedaan performa mencerminkan kompleksitas komputasi masing-masing *model*.

6. *Maintenance*

   Tahap *maintenance* dilakukan secara berkelanjutan berdasarkan hasil pengujian dan umpan balik dari tahap sebelumnya. Aktivitas pada tahap ini mencakup perbaikan minor, penyesuaian konfigurasi sistem, serta optimalisasi performa tanpa mengubah arsitektur utama. Tahap ini memastikan sistem AIRA tetap stabil dan dapat digunakan kembali untuk iterasi pengujian atau pengembangan lanjutan di masa depan.

3.1.1 Model Software Development Life Cycle

Model Software Development Life Cycle (SDLC) yang digunakan dalam penelitian ini adalah Agile SDLC. Model Agile dipilih karena karakteristik pengembangan sistem AIRA yang bersifat iteratif dan eksperimental, khususnya dalam integrasi model AI dan evaluasi performa *backend*. Pengembangan sistem deteksi kanker berbasis AI memerlukan fleksibilitas dalam penyesuaian arsitektur, konfigurasi model, serta skenario pengujian, yang sulit dicapai apabila menggunakan model SDLC linear seperti Waterfall.

Pendekatan Agile memungkinkan proses pengembangan dilakukan secara bertahap melalui siklus iterasi yang berulang, sehingga setiap komponen sistem dapat diuji dan disempurnakan secara bertahap berdasarkan hasil evaluasi sebelumnya. Hal ini sangat relevan dalam konteks penelitian ini, di mana perubahan

pada konfigurasi *API gateway*, arsitektur *backend*, maupun skenario *load testing* dapat terjadi selama proses penelitian berlangsung.

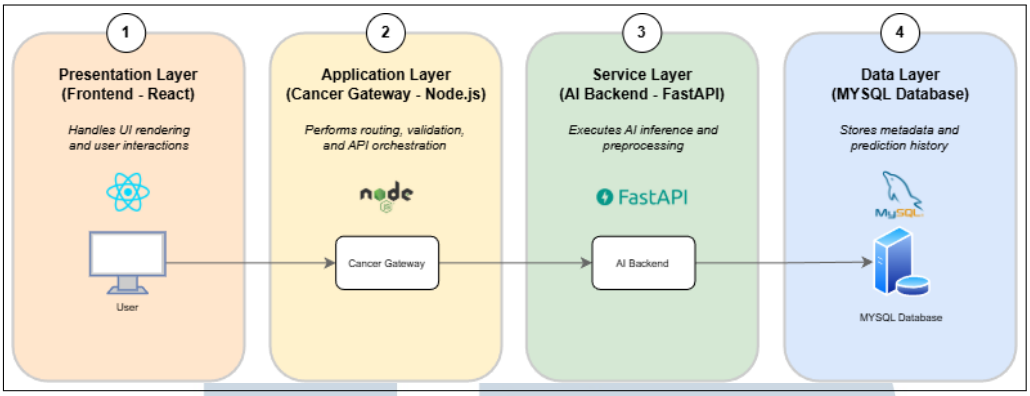
Kelebihan utama Agile SDLC dalam penelitian ini adalah kemampuannya untuk mendukung pengembangan yang adaptif, mempercepat umpan balik dari hasil pengujian performa, serta meminimalkan risiko kesalahan desain arsitektur sejak tahap awal. Dengan demikian, penggunaan Agile SDLC dinilai sesuai untuk mendukung tujuan penelitian dalam mengevaluasi performa dan efisiensi sistem AIRA secara sistematis dan terukur.

3.2 Perancangan Arsitektur Backend

Perancangan arsitektur *backend* dilakukan untuk memastikan sistem AIRA mampu menangani proses *machine learning inference* secara terstruktur, efisien, dan mudah dikembangkan. Pada tahap ini, fokus utama diarahkan pada pemisahan tanggung jawab antar-komponen sistem, pengelolaan alur data, dan implementasi struktur *database* pada sistem. Arsitektur dirancang dengan mempertimbangkan karakteristik *ML workload* yang bersifat *compute-intensive* dan berbeda dari layanan web konvensional. Hasil perancangan ini menjadi dasar implementasi dan evaluasi performa yang dibahas pada bab-bab selanjutnya.

3.2.1 Layered Architecture Design

Arsitektur sistem AIRA dirancang menggunakan pendekatan *layered architecture* yang terdiri dari empat *layer* utama. Presentation layer berupa *frontend* yang menyediakan antarmuka pengguna untuk mengunggah data genomik dan menampilkan hasil prediksi. Application layer diisi oleh Cancer Gateway yang menangani *routing request*, validasi input, *rate limiting*, serta komunikasi dengan basis data untuk manajemen metadata. Service layer berupa AI Backend yang bertanggung jawab untuk *preprocessing* data, *loading models*, melakukan *inference*, dan mengembalikan hasil prediksi. Data layer berupa MySQL database yang menyimpan konfigurasi model, jenis kanker, serta riwayat prediksi.

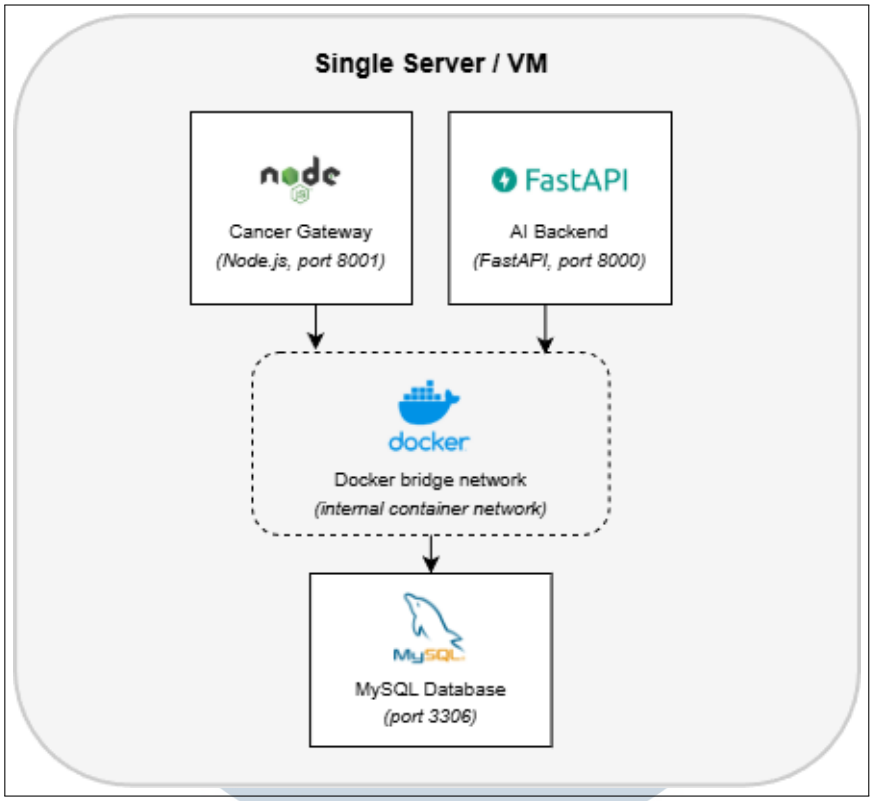


Gambar 3.2. Layered Architecture Design of AIRA

Pemisahan ke dalam empat layer yang ditampilkan pada Gambar 3.2 ini memberikan *separation of concerns* yang jelas antara antarmuka pengguna, logika aplikasi, proses inferensi AI, dan penyimpanan data. Pendekatan tersebut memudahkan proses pengembangan dan pemeliharaan karena setiap layer dapat dikembangkan atau dimodifikasi secara relatif independen. Selain itu, pemisahan Application layer (Cancer Gateway) dan Service layer (AI Backend) mendukung tujuan penelitian ini, yaitu mengevaluasi performa arsitektur backend berbasis API gateway untuk *ML inference workload* tanpa mengganggu lapisan presentasi maupun lapisan data.

3.2.2 Deployment Strategy

Strategi *deployment* pada sistem AIRA dirancang untuk mendukung pengujian performa *backend* secara terkontrol dan efisien. Pendekatan yang digunakan menekankan kesederhanaan arsitektur, minimisasi *latency*, serta kemudahan pengelolaan layanan selama fase penelitian. Oleh karena itu, sistem diimplementasikan menggunakan Docker *containerization* dengan pendekatan *co-location* dalam satu *server*.



Gambar 3.3. Deployment Architecture Sistem AIRA dengan Docker

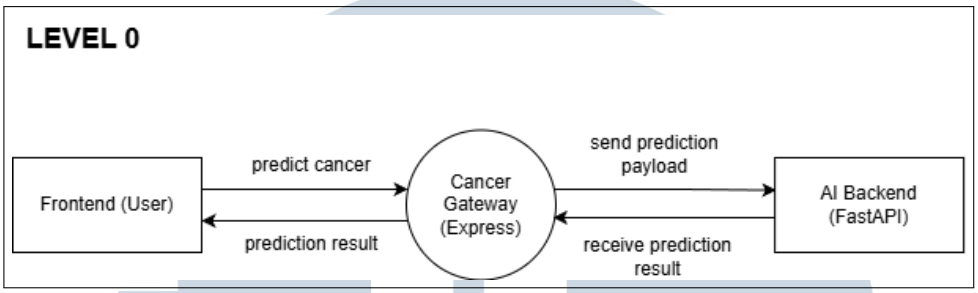
Pada Gambar 3.3 ini, terdapat dua *service* utama yaitu *Cancer Gateway* dan AI *Backend* yang dikemas dalam *container* terpisah untuk memastikan isolasi proses dan independensi layanan. Basis data MySQL dijalankan langsung pada VM tanpa *containerization*, namun tetap berada pada *server* fisik yang sama untuk mengurangi *network overhead*. Komunikasi antar-*container* difasilitasi melalui Docker *bridge network* yang memungkinkan pertukaran data melalui *internal network* dengan *latency* minimal. Seluruh layanan didefinisikan dan dijalankan secara konsisten menggunakan Docker Compose, sehingga memudahkan proses *deployment*, *maintenance*, dan replikasi lingkungan pengujian.

3.2.3 Data Flow Diagram

1. DFD Level 0

DFD *Level 0* digunakan untuk memberikan gambaran umum alur data dan batasan sistem AIRA secara keseluruhan. Diagram ini memfokuskan pada interaksi antara pengguna sebagai entitas eksternal dengan sistem inti tanpa menampilkan detail proses internal. Tujuan utama penyusunan DFD *Level 0* adalah

untuk menunjukkan bagaimana data mengalir dari pengguna hingga menghasilkan keluaran prediksi.

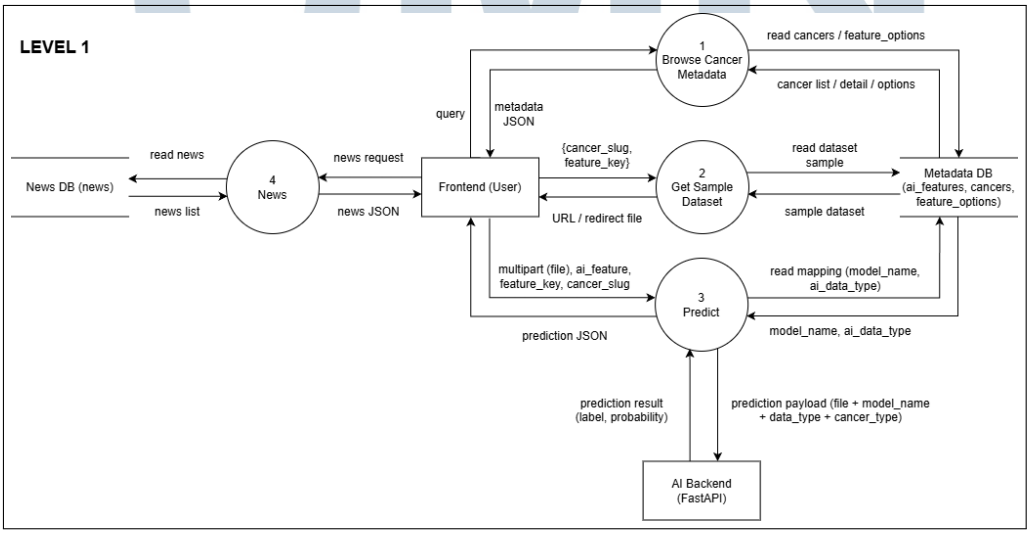


Gambar 3.4. Data Flow Diagram Level 0

Pada Gambar 3.4 ditampilkan DFD *Level 0* yang dimana sistem AIRA direpresentasikan sebagai satu proses utama yang menerima input berupa berkas data genomik dari pengguna melalui *frontend*. Data tersebut diteruskan ke *Cancer Gateway* untuk diproses dan kemudian dikirim ke AI *Backend* (FastAPI) guna melakukan proses *inference*. Output dari sistem berupa hasil prediksi kanker beserta nilai probabilitas yang dikembalikan kepada pengguna. Diagram ini menegaskan batas sistem dan alur pertukaran data antara entitas eksternal dan proses inti AIRA.

2. DFD Level 1

DFD *Level 1* disusun untuk memberikan representasi alur data yang lebih rinci dibandingkan *Level 0*. Diagram ini memecah proses utama AIRA menjadi beberapa komponen fungsional yang saling berinteraksi. Dengan pendekatan ini, setiap peran sistem dapat dianalisis secara lebih terstruktur.

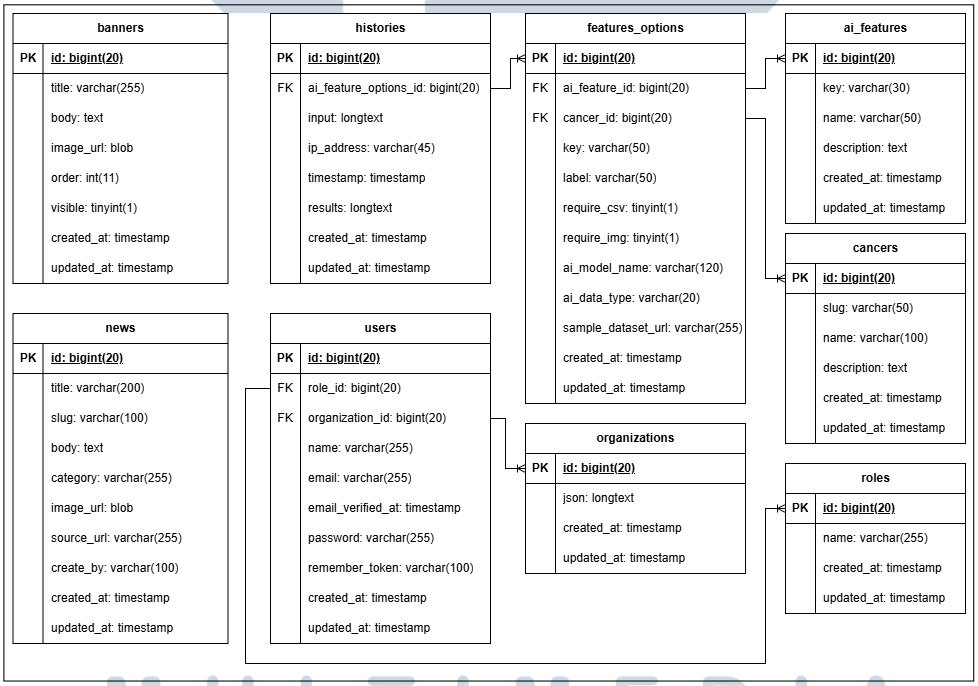


Gambar 3.5. Data Flow Diagram Level 1

Pada DFD *Level 1* yang ditampilkan pada Gambar 3.5, sistem dibagi menjadi *Cancer Gateway*, AI *Backend*, dan *Metadata/News Database*. *Cancer Gateway* berfungsi menangani validasi input, *routing request*, pengambilan metadata, penyajian sampel dataset, serta pengiriman *payload* prediksi ke AI *Backend*. AI *Backend* melakukan *preprocessing* data, pemetaan *model*, dan eksekusi *inference* sebelum mengembalikan hasil prediksi. Sementara itu, *database* menyediakan data pendukung seperti konfigurasi *model*, fitur, contoh dataset, dan konten berita yang dibutuhkan oleh sistem.

3.2.4 Database Schema

Perancangan database pada sistem AIRA bertujuan untuk menyimpan metadata kanker, konfigurasi model, opsi fitur yang tersedia, serta riwayat permintaan prediksi. Selain itu, skema juga memuat sejumlah tabel tambahan untuk kebutuhan CMS seperti manajemen berita, banner, dan pengguna sistem.



Gambar 3.6. AIRA Database Schema

Gambar 3.6 menampilkan struktur lengkap database yang digunakan pada penelitian ini. Penelitian ini berfokus pada empat tabel inti yang secara langsung mendukung proses inferensi AI, yaitu `cancers`, `ai_features`, `features_options`, dan `histories`. Penjelasan masing-masing tabel adalah sebagai berikut:

1. Tabel `cancers`

   Tabel ini menyimpan daftar jenis kanker yang dapat diprediksi oleh sistem, seperti kanker prostat dan kanker payudara. Setiap entri memuat *slug*, nama kanker, dan deskripsi singkat yang digunakan sebagai metadata pada antarmuka pengguna maupun proses backend.

2. Tabel `ai_features`

   Tabel ini berisi daftar tipe fitur biologis yang digunakan oleh model AI, seperti *gene expression*, DNA *methylation*, dan *microRNA*. Kolom *key* dan nama fitur digunakan untuk mengelompokkan model berdasarkan karakteristik data yang dipakai.

3. Tabel `features_options`

   Tabel ini merupakan komponen kunci dalam sistem AIRA karena menghubungkan jenis kanker (`cancer_id`), jenis fitur AI (`ai_feature_id`), serta konfigurasi model yang tersedia. Informasi seperti `key`, `label`, `ai_model_name`, `ai_data_type`, dan `sample_dataset_url` digunakan oleh Cancer Gateway untuk menentukan model prediksi dan *sample dataset* yang sesuai.

4. Tabel `histories`

   Tabel ini menyimpan riwayat permintaan prediksi yang dilakukan pengguna. Kolom-kolom di dalamnya mencatat referensi ke `features_options`, data input dalam bentuk teks, alamat IP, waktu permintaan, serta hasil prediksi yang dikembalikan oleh AI Backend. Informasi ini berfungsi sebagai dasar untuk kebutuhan audit, *logging*, dan analisis penggunaan sistem di masa mendatang.

Selain tabel inti tersebut, skema basis data AIRA juga mencakup sejumlah tabel tambahan yang digunakan untuk mendukung fungsionalitas CMS dan manajemen pengguna. Tabel-tabel ini tidak berhubungan langsung dengan proses inferensi AI, namun tetap ditampilkan dalam skema untuk memberikan gambaran menyeluruh mengenai lingkungan aplikasi. Penjelasan ringkas tabel-tabel tersebut adalah sebagai berikut:

1. Tabel `news` menyimpan artikel atau berita yang ditampilkan pada aplikasi.

2. Tabel `banners` digunakan untuk manajemen gambar promosi atau banner informatif.

3. Tabel `users` menyimpan akun pengguna admin beserta informasi organisasi dan autentikasinya.

4. Tabel `roles` mendefinisikan peran pengguna untuk keperluan otorisasi.

5. Tabel `organizations` menyimpan metadata organisasi yang terkait dengan pengguna admin.

3.3 List Service Development

Pada penelitian ini, *Cancer Gateway* menyediakan sejumlah *RESTful endpoints* yang digunakan oleh *frontend* untuk berinteraksi dengan sistem AIRA. Daftar layanan utama yang dikembangkan ditunjukkan pada Tabel 3.1.

Tabel 3.1. Daftar Layanan Backend (*Service Endpoints*)

Service	**Endpoint**	**Deskripsi**
Get Cancers	/cancers?ai_feature={fitur_ai}	Mengambil daftar jenis kanker yang dapat diprediksi berdasarkan *feature* tertentu.
Cancer Detail	/cancers/:slug?ai_feature={fitur_ai}	Mengambil detail jenis kanker, termasuk deskripsi, gejala umum, dan model AI yang digunakan.
Feature Option	/cancers/:slug/feature-options?ai_feature={fitur_ai}	Mengambil daftar opsi fitur (tipe dataset dan jumlah fitur), contoh: `GENE-35`.
Sample Dataset	/cancers/:slug/feature-options/:feature_key/sample-dataset?ai_feature={fitur_ai}	Mengunduh contoh berkas CSV sebagai *sample dataset* sesuai *feature key*.
Get News	/news	Mengambil daftar berita terkait sistem AIRA.
News Detail	/news/{news_id}	Mengambil detail berita berdasarkan *ID*.
Predict Cancer	/cancers/:slug/predict (multipart)	Mengirim berkas CSV untuk melakukan prediksi kanker menggunakan model sesuai *feature key* dan jenis kanker.

Layanan-layanan yang tercantum pada Tabel 3.1 merepresentasikan fungsi utama yang disediakan oleh *Cancer Gateway* sebagai perantara antara *frontend* dan *AI Backend*. Setiap *endpoint* dirancang untuk mendukung alur penggunaan sistem AIRA, mulai dari penyajian informasi kanker hingga proses *machine learning inference* melalui pengiriman berkas dataset. Daftar layanan ini juga menjadi dasar penentuan *endpoint* yang diuji pada skenario *load testing* dan *stress testing* di Bab IV. Dengan demikian, evaluasi performa sistem difokuskan pada *endpoint* yang secara langsung merepresentasikan beban kerja nyata pada sistem.

3.4 Skenario Load Test dan Stress Testing

Pengujian *load test* dan *stress testing* dilakukan untuk mengevaluasi bagaimana arsitektur *backend* AIRA berperforma ketika menerima beban permintaan yang meningkat. Fokus utama pengujian adalah mengamati *response time*, *throughput*, *error rate*, dan utilisasi *resource* ketika sistem mendekati dan melewati kapasitas normalnya, sehingga dapat diidentifikasi batas kemampuan dan indikasi *bottleneck*.

Metrik yang digunakan pada pengujian meliputi:

- *Response time*: waktu yang dibutuhkan sejak *request* dikirim hingga *response* diterima. Analisis difokuskan pada nilai rata-rata, minimum, maksimum, serta distribusi persentil pada beberapa skenario pengujian.
- *Throughput*: jumlah *requests per second* (RPS) yang dapat diproses sistem selama periode pengujian.
- *Error rate*: persentase *request* yang gagal (HTTP status 4xx/5xx) pada tiap skenario pengujian sebagai indikator reliabilitas sistem di bawah beban.

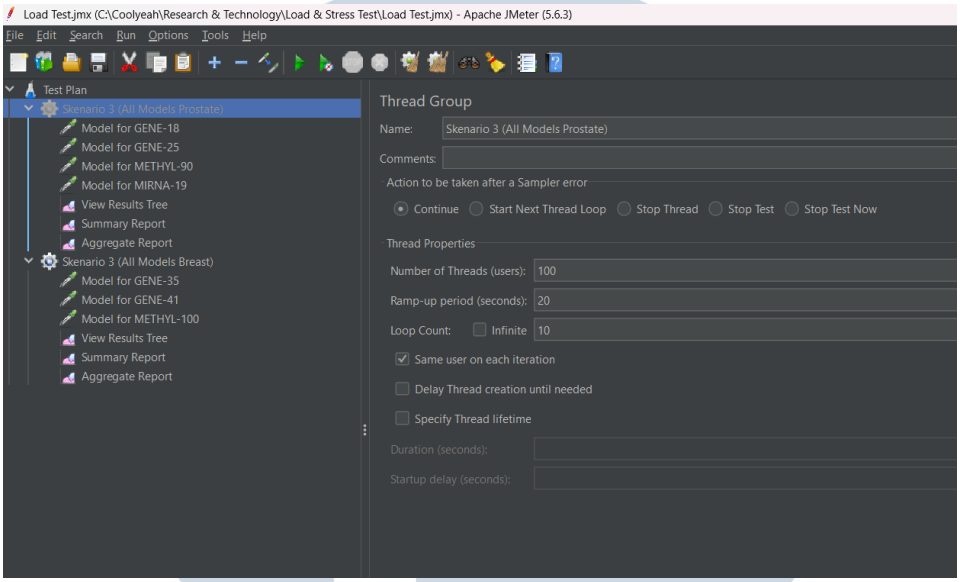
3.4.1 Lingkungan Pengujian

Lingkungan *testing* disiapkan dengan spesifikasi *hardware* dan konfigurasi yang memadai untuk menjalankan sistem dan *load testing* secara bersamaan. Spesifikasi *hardware* mencakup prosesor, kapasitas RAM, dan *storage* pada *server* yang digunakan untuk men-*deploy* sistem AIRA dan menjalankan Apache JMeter. Konfigurasi jaringan menggunakan *localhost* atau *local network* untuk meminimalkan variabilitas *network latency* eksternal yang dapat mempengaruhi hasil pengukuran. Apache JMeter dipilih sebagai *tool* utama untuk *load testing* karena memiliki antarmuka yang *user-friendly* dan kemampuan *reporting* yang cukup lengkap. *Monitoring tools* tambahan seperti Docker *stats* dan *htop* digunakan untuk mengumpulkan data utilisasi *resource* selama *testing*.

3.4.2 Konfigurasi dan Prosedur Pengujian Menggunakan Apache JMeter

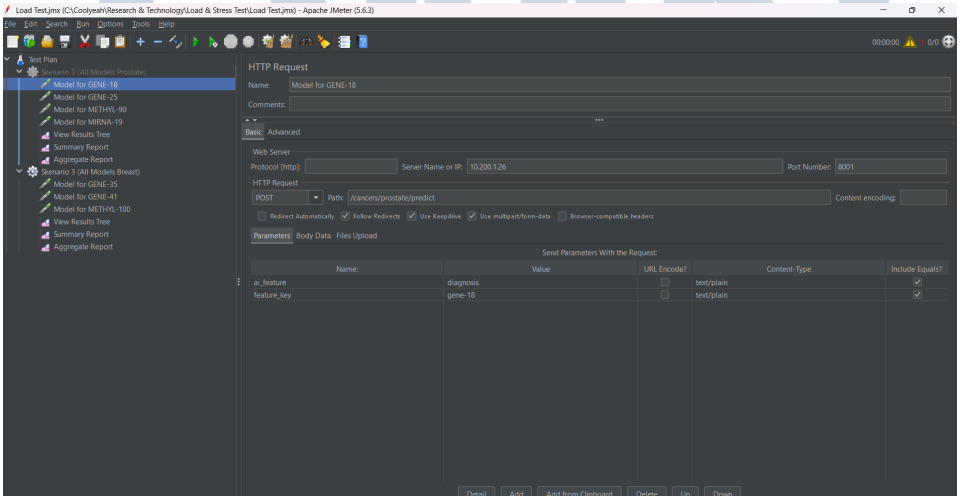
Pengujian performa sistem AIRA dilakukan menggunakan Apache JMeter dengan memanfaatkan komponen *Thread Group* dan *HTTP Request Sampler*. Thread Group digunakan untuk mensimulasikan jumlah pengguna simultan

(*concurrent users*), ramp-up period, serta jumlah iterasi pengujian sesuai dengan skenario *load testing* dan *stress testing* yang ditetapkan.



Gambar 3.7. Konfigurasi Thread Group pada Apache JMeter untuk Simulasi Beban Pengguna

Pada Gambar 3.7 ditunjukkan konfigurasi Thread Group yang digunakan untuk mengatur jumlah pengguna simultan dan pola beban. Setiap pengguna mengirimkan permintaan HTTP secara paralel ke sistem backend untuk mensimulasikan kondisi penggunaan nyata.



Gambar 3.8. Konfigurasi HTTP Request Sampler untuk Endpoint Prediksi Kanker

Selanjutnya, setiap Thread Group dikonfigurasikan dengan satu atau lebih

HTTP Request Sampler sebagaimana ditunjukkan pada Gambar 3.8. HTTP Request ini merepresentasikan endpoint prediksi kanker dengan metode POST, lengkap dengan parameter *ai_feature* dan *feature_key* yang sesuai dengan model AI yang diuji. Pendekatan ini memastikan bahwa pengujian dilakukan pada endpoint yang sama dengan yang digunakan oleh aplikasi *frontend*.

Hasil pengujian dikumpulkan menggunakan *Aggregate Report* dan *Summary Report* pada Apache JMeter, kemudian dianalisis dalam bentuk tabel dan grafik untuk mengevaluasi *response time*, *throughput*, dan *error rate*.

3.4.3 Skenario Pengujian Seluruh API

Skenario pertama dirancang untuk mengevaluasi stabilitas seluruh layanan *REST API* pada sistem AIRA, yang mencakup layanan GET (metadata, berita, dan fitur) serta layanan POST (prediksi). Pengujian dilakukan pada dua tingkat beban umum untuk merepresentasikan kondisi penggunaan normal dan beban menengah. Fokus utama pengujian ini adalah memastikan seluruh *endpoint* dapat merespons permintaan secara konsisten sebelum dilakukan pengujian yang lebih terfokus pada *endpoint* prediksi.

Tabel 3.2. Skenario Pengujian Seluruh API (GET dan POST)

Skenario	Concurrent Users	Total Requests	Ramp-up Time	Durasi
Low	10	100	5 detik	1 menit
High	100	500	10 detik	2 menit

Tabel 3.2 menunjukkan konfigurasi pengujian untuk seluruh layanan API pada sistem AIRA. Skenario *Low* digunakan sebagai *baseline* untuk memverifikasi bahwa seluruh *endpoint* GET dan POST dapat berfungsi dengan baik pada kondisi beban ringan. Sementara itu, skenario *High* merepresentasikan kondisi beban menengah dengan jumlah pengguna yang lebih besar untuk mengamati stabilitas sistem secara keseluruhan sebelum dilakukan pengujian yang lebih spesifik pada *endpoint* prediksi.

3.4.4 Skenario Load Test dan Stress Test untuk Endpoint Prediksi

Skenario kedua berfokus pada *endpoint* prediksi kanker (POST multipart), yang merupakan komponen inti dari penelitian ini. Pengujian dilakukan secara bertahap melalui skenario *load test* dan *stress test* untuk mengidentifikasi batas

performa sistem ketika menangani *ML inference workload*. Secara umum, skenario dibagi menjadi beberapa tingkat beban sebagai berikut:

- *Baseline load*: mensimulasikan penggunaan normal dengan jumlah *concurrent users* dan total *requests* yang relatif rendah untuk memperoleh *baseline performance*.
- *Increased load*: mensimulasikan kondisi beban menengah dengan peningkatan jumlah *concurrent users* dan total *requests* untuk melihat bagaimana sistem beradaptasi terhadap beban yang lebih tinggi.
- *Stress load*: mensimulasikan kondisi *stress* dengan jumlah *concurrent users* dan total *requests* yang lebih besar untuk mengidentifikasi batas kapasitas dan titik ketika sistem mulai mengalami degradasi performa dan peningkatan *error rate*.

Setiap skenario direncanakan dengan parameter umum seperti jumlah *concurrent users*, total *requests*, *ramp-up time*, dan durasi pengujian. Parameter rinci untuk setiap skenario akan dirangkum dalam tabel yang mencakup tujuan masing-masing skenario.

Tabel 3.3. Skenario Load Test dan Stress Test Endpoint Prediksi Model Prostate

Skenario	Concurrent Users	Total Requests	Ramp-up Time	Durasi
Load 1	10	100	5 detik	1 menit
Load 2	20	200	5 detik	1 menit
Load 3	50	500	10 detik	2 menit
Load 4	100	1000	15 detik	3 menit
Stress 1	100	1500	10 detik	2 menit
Stress 2	200	2500	15 detik	2 menit
Stress 3	500	4000	20 detik	3 menit
Stress 4	1000	6000	30 detik	3 menit

Tabel 3.3 menunjukkan konfigurasi skenario pengujian untuk *endpoint* prediksi model kanker prostat. Pengujian dimulai dari beban ringan dengan 10 *concurrent users* sebagai *baseline*, kemudian ditingkatkan secara bertahap hingga kondisi *stress* dengan 1000 pengguna bersamaan. Konfigurasi ini digunakan untuk mengamati perubahan *response time*, *throughput*, dan *error rate* seiring meningkatnya beban permintaan inferensi.

Tabel 3.4. Skenario Load Test dan Stress Test Endpoint Prediksi Model Breast

Skenario	Concurrent Users	Total Requests	Ramp-up Time	Durasi
Load 1	10	100	5 detik	1 menit
Load 2	20	200	5 detik	1 menit
Load 3	50	500	10 detik	2 menit
Load 4	100	1000	15 detik	3 menit
Stress 1	100	1500	10 detik	2 menit
Stress 2	200	2500	15 detik	2 menit
Stress 3	500	4000	20 detik	3 menit
Stress 4	1000	6000	30 detik	3 menit

Tabel 3.4 menyajikan skenario pengujian yang sama untuk *endpoint* prediksi model kanker payudara. Penggunaan parameter yang identik dengan pengujian model prostat bertujuan untuk memastikan konsistensi skenario sehingga hasil pengujian kedua model dapat dibandingkan secara langsung. Perbedaan performa yang muncul pada tahap analisis selanjutnya dengan demikian dapat dikaitkan terutama dengan kompleksitas komputasi model, bukan dengan variasi konfigurasi pengujian.

3.4.5 Skenario Perbandingan Model Prostate dan Breast

Selain dua skenario sebelumnya, penelitian ini juga menyertakan satu skenario tambahan yang berfokus pada perbandingan performa antar model. Tujuan skenario ini adalah untuk mengetahui model mana yang memiliki beban komputasi paling ringan hingga paling berat, baik pada kelompok kanker prostat maupun kanker payudara, ketika dijalankan dengan konfigurasi *workload* yang sama. Dengan demikian, hasil pengujian tidak hanya menunjukkan performa arsitektur secara umum, tetapi juga memberikan gambaran relatif mengenai karakteristik masing-masing model yang di-*deploy*.

Seluruh model diuji dengan konfigurasi *concurrent users* yang sama, yaitu 100 pengguna, *ramp-up time* selama 20 detik, dan *loop count* sebanyak 10. Setiap *virtual user* akan mengirimkan permintaan prediksi berulang sesuai *loop count*, sehingga diperoleh jumlah sampel yang cukup untuk setiap model. Pemilihan 100 *concurrent users* dimaksudkan sebagai titik tengah antara beban normal dan beban mendekati *stress*, sehingga hasilnya masih relevan untuk skenario penggunaan nyata, tetapi cukup menonjol untuk membedakan performa antar model. *Ramp-up* 20 detik dipilih agar kenaikan beban tidak terlalu mendadak, namun tetap memberikan tekanan yang konsisten pada *backend*.

Tabel 3.5. Skenario Perbandingan Model Prostate dan Breast

No	Model	Concurrent Users	Loop Count	Ramp-up Time
1	Prostate – GENE-18	100	10	20 detik
2	Prostate – GENE-25	100	10	20 detik
3	Prostate – METHYL-90	100	10	20 detik
4	Prostate – MIRNA-19	100	10	20 detik
5	Breast – GENE-35	100	10	20 detik
6	Breast – GENE-41	100	10	20 detik
7	Breast – METHYL-100	100	10	20 detik

Hasil dari skenario yang ditampilkan pada Tabel 3.5 ini akan dianalisis pada Bab selanjutnya bersama dengan skenario lainnya, dengan fokus pada perbandingan *average response time*, *throughput*, dan *error rate* antar model. Dari sana dapat ditarik kesimpulan mengenai model mana yang paling ringan dan paling berat secara komputasi, serta implikasinya terhadap kapasitas sistem ketika jumlah model yang di-*serve* semakin bertambah.

3.4.6 Persiapan Data Uji dan Analisis

Data uji disiapkan dalam bentuk sampel CSV data genomik yang merepresentasikan berbagai tipe data yang didukung oleh sistem AIRA, seperti *gene expression*, DNA *methylation*, dan *microRNA*. Variasi ukuran *payload* digunakan untuk mengamati pengaruh ukuran data input terhadap *response time* dan *throughput*. Format dan struktur data mengikuti spesifikasi yang diharapkan oleh AI *Backend*.

Analisis hasil pengujian dilakukan dengan menghitung statistik deskriptif untuk setiap skenario, memvisualisasikan hasil dalam bentuk grafik (misalnya *line chart* untuk *response time over time* dan *bar chart* untuk *throughput*), serta mengamati pola *error rate* dan utilisasi *resource*. Hasil pengamatan kemudian dibandingkan secara kualitatif dengan rentang *response time* dari literatur untuk menilai apakah performa sistem AIRA berada pada level yang masih *acceptable*.