LAPORAN PROYEK AKHIR

MATA KULIAH TOPIK KHUSUS TEKNIK BIOMEDIS I

SKINALYZE: APLIKASI WEB BERBASIS DEEP LEARNING UNTUK KLASIFIKASI PENYAKIT KULIT

Disusun oleh:

1. Aisha Nabilla Niko Amani (21/472993/TK/52116)
2. Irfan Maulana Marantika (21/473459/TK/52180)
3. Willybrodus Andhika Budikusuma (21/473598/TK/52201)
4. Josephine Florina Wijaya (21/478572/TK/52737)
5. Ignatius Gilbert Wicaksana (21/480202/TK/52985)

DEPARTEMEN TEKNIK ELEKTRO DAN TEKNOLOGI INFORMASI

FAKULTAS TEKNIK UNIVERSITAS GADJAH MADA

YOGYAKARTA

2024

DAFTAR ISI

BAB I
PENDAHULUAN

a. Latar Belakang

Penyakit kulit, meskipun sering kali dianggap masalah kesehatan yang ringan, sebenarnya dapat memberikan dampak besar bagi penderita, baik secara fisik maupun psikologis [1]. Kondisi seperti jerawat, eksim, psoriasis, hingga kanker kulit, merupakan beberapa contoh penyakit kulit yang sering terjadi di masyarakat. Namun, banyak orang yang mengalami kesulitan dalam mengidentifikasi gejala awal penyakit kulit, terutama ketika penyakit tersebut tidak menunjukkan tanda yang jelas atau bahkan menyerupai penyakit lain. Hal ini sering kali membuat penderita menunda untuk mencari pengobatan, yang pada akhirnya dapat memperburuk kondisi kulit mereka.

Proses diagnosis penyakit kulit secara tradisional bergantung pada konsultasi dengan dokter spesialis kulit yang memerlukan biaya dan waktu yang tidak sedikit. Selain itu, banyak daerah yang kekurangan tenaga medis terlatih, sehingga akses untuk mendapatkan diagnosis yang tepat dan cepat menjadi terbatas. Kondisi ini sering kali menghambat penanganan yang cepat, sehingga pada akhirnya berisiko bagi kesehatan kulit jangka panjang. Sebagai contoh, dalam kasus kanker kulit, deteksi dini dapat sangat menentukan kesuksesan pengobatan dan mengurangi tingkat keparahan penyakit.

Selain masalah akses, tantangan lain adalah keterbatasan dalam akurasi diagnosis. Diagnosis yang bergantung pada keahlian manusia seringkali dipengaruhi oleh faktor seperti waktu yang terbatas, kesalahan manusia, atau ketidaktahuan tentang perkembangan penyakit. Oleh karena itu, dibutuhkan solusi yang lebih cepat, lebih terjangkau, dan lebih akurat untuk membantu mendiagnosis penyakit kulit secara lebih efisien.

Kemajuan teknologi, terutama di bidang kecerdasan buatan (AI) dan pembelajaran mesin (ML), memberikan solusi untuk tantangan ini. Teknologi ini memungkinkan pengembangan perangkat analisis kulit yang dapat memberikan diagnosis secara otomatis dan cepat. Beberapa perangkat telah berhasil digunakan dalam industri kecantikan untuk menganalisis kondisi kulit [2], dan semakin banyak aplikasi kesehatan yang menggunakan teknologi ini untuk deteksi penyakit kulit secara dini. Keuntungan

utama dari teknologi ini adalah kemampuannya untuk menganalisis gambar kulit dengan akurasi tinggi, memberikan hasil dalam waktu singkat, serta dapat diakses oleh pengguna melalui ponsel pintar mereka, bahkan di daerah yang kekurangan tenaga medis terlatih.

Dengan perkembangan ini, aplikasi seperti Skinalyze hadir sebagai solusi yang mudah diakses, terjangkau, dan efektif untuk membantu masyarakat dalam mendeteksi penyakit kulit sejak dini. Aplikasi ini bertujuan untuk menjembatani kesenjangan yang ada dengan memberikan alternatif diagnosis yang praktis dan dapat diakses kapan saja dan di mana saja, memungkinkan pengguna untuk mengambil memulai pencegahan lebih awal sebelum kondisi kulit mereka memburuk.

b. Rumusan Masalah

Pengembangan aplikasi ini berangkat dari beberapa permasalahan utama:

1. Akses terbatas ke dokter spesialis kulit terutama di daerah dengan sumber daya kesehatan yang minim
2. Biaya diagnosis yang tinggi, sehingga sulit untuk dijangkau oleh sebagian masyarakat
3. Kesulitan dalam memperoleh diagnosis yang cepat, terutama untuk kasus penyakit kulit yang memerlukan perhatian segera
4. Ketergantungan pada metode diagnosis tradisional yang memiliki potensi kesalahan akibat faktor manusia

c. Solusi yang Ditawarkan

Pada laporan ini, Skinalyze diperkenalkan sebagai solusi inovatif untuk mendeteksi penyakit kulit secara otomatis dengan menggunakan teknologi berbasis kecerdasan buatan. Aplikasi ini dirancang untuk membantu mendiagnosis penyakit kulit secara otomatis. Aplikasi ini menawarkan beberapa keunggulan utama:

1. Cepat dan akurat: Skinalyze dapat menganalisis foto kulit dan memberikan hasil diagnosis dalam waktu singkat dengan akurat
2. Terjangkau: Solusi ini memungkinkan pengguna mendapat diagnosis dengan biaya yang lebih rendah

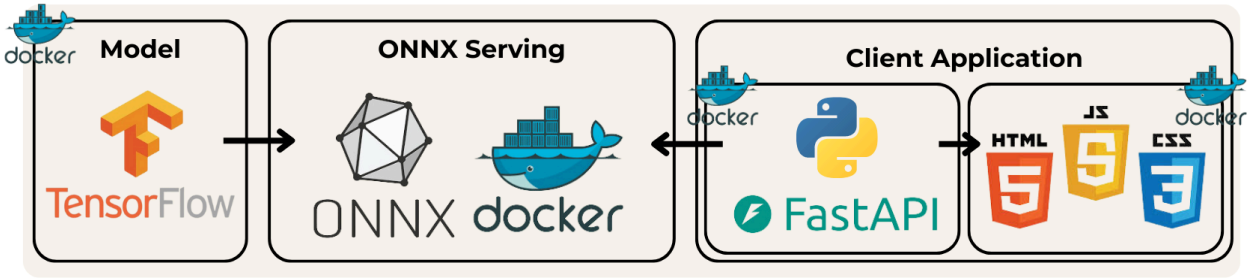
3. Mudah diakses: Aplikasi ini dapat digunakan melalui ponsel sehingga mudah digunakan dan pengguna di daerah terpencil sekalipun dapat memanfaatkannya

d. Tujuan Solusi

Tujuan dari aplikasi ini adalah:

1. Mengembangkan aplikasi Skinalyze yang mampu mendeteksi penyakit kulit secara otomatis menggunakan teknologi kecerdasan buatan
2. Memberikan solusi yang cepat, akurat dan terjangkau bagi masyarakat luas
3. Mempermudah pengguna dalam memantau kondisi kulit secara mandiri
4. Meningkatkan kesadaran masyarakat akan pentingnya diagnosis dini untuk penyakit sehingga mengurangi risiko jangka panjang

BAB II
PEMODELAN PERMASALAHAN

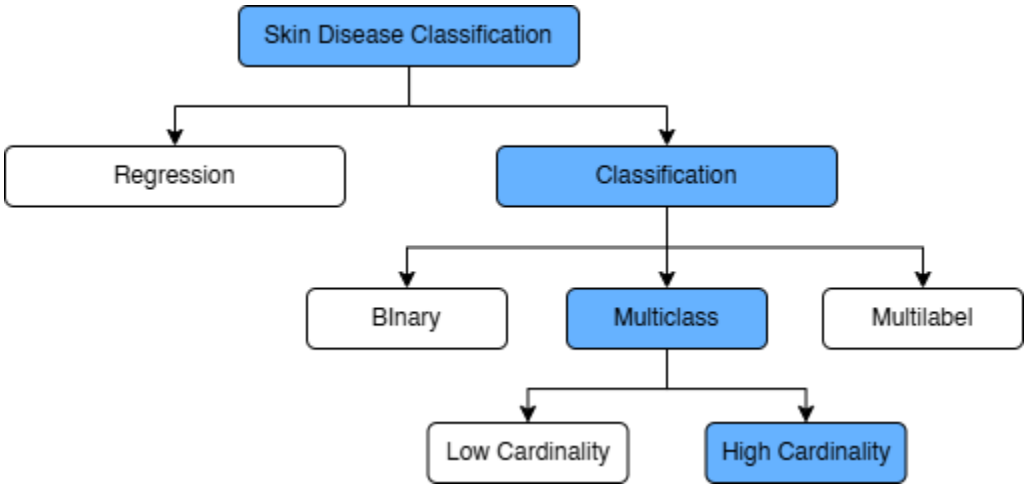


Gambar 2.1 Arsitektur Microservices

Permasalahan yang dihadapi dalam pengembangan proyek ini berkaitan dengan efisiensi deteksi penyakit kulit menggunakan teknologi berbasis gambar. Pemodelan permasalahan ini melibatkan identifikasi dan pengintegrasian teknologi machine learning untuk mendukung klasifikasi penyakit kulit secara otomatis. Proses ini mencakup penerimaan gambar dari pengguna, pengolahan data menggunakan model deep learning untuk menghasilkan diagnosis, dan pengembalian hasil kepada pengguna melalui antarmuka yang intuitif. Aplikasi Skinalyze juga menghadapi tantangan integrasi antara komponen frontend, backend, dan model machine learning dalam arsitektur microservices untuk memastikan kinerja aplikasi yang optimal.

a. Framing Machine Learning Problem

Pada bagian ini, dijelaskan alur dalam pemilihan model machine learning untuk *Skin Disease Classification*. Untuk memudahkan pemahaman, berikut ini adalah grafik yang menjelaskan alur pemilihan model machine learningnya.

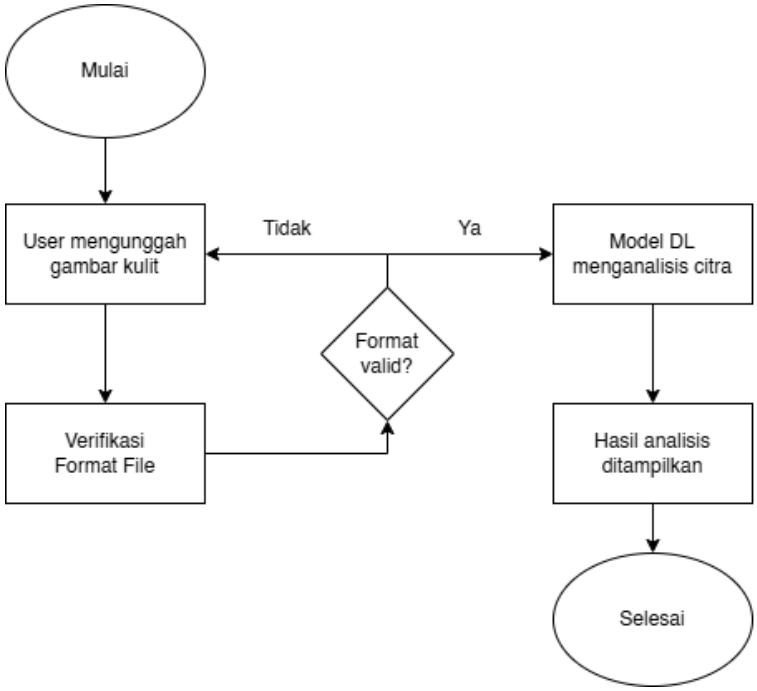


Gambar 2.2 *Framing ML Problem*

Pada gambar 2.2 menggambarkan framing problem dalam machine learning ini, topik utama yang diangkat adalah *Skin Disease Classification*. Dalam skema ini, proses dimulai dengan pemilihan jenis masalah, yang terbagi menjadi dua kategori utama, yaitu regresi dan klasifikasi. Untuk proyek ini, fokus utama terletak pada kategori klasifikasi, yang selanjutnya dibagi lagi menjadi tiga jenis: *binary classification*, *multiclass classification*, dan *multilabel classification*. Setelah mempertimbangkan kebutuhan dan karakteristik data, dipilihlah pendekatan *multiclass classification*. Selanjutnya, pada kategori *multiclass*, masalah ini dibagi lagi menjadi dua subkategori, yaitu *low cardinality* dan *high cardinality*. Berdasarkan kompleksitas data dan jumlah kelas yang ada, dipilihlah pendekatan *high cardinality*, yang memungkinkan klasifikasi dengan jumlah kelas yang lebih banyak.

b. *Flowchart Diagram*

Pada bagian ini, dijelaskan alur penggunaan aplikasi “Skinalyze”. Untuk memudahkan pemahaman, berikut ini adalah flowchart yang menggambarkan tahapan-tahapan yang dilalui dalam penggunaan aplikasi tersebut.

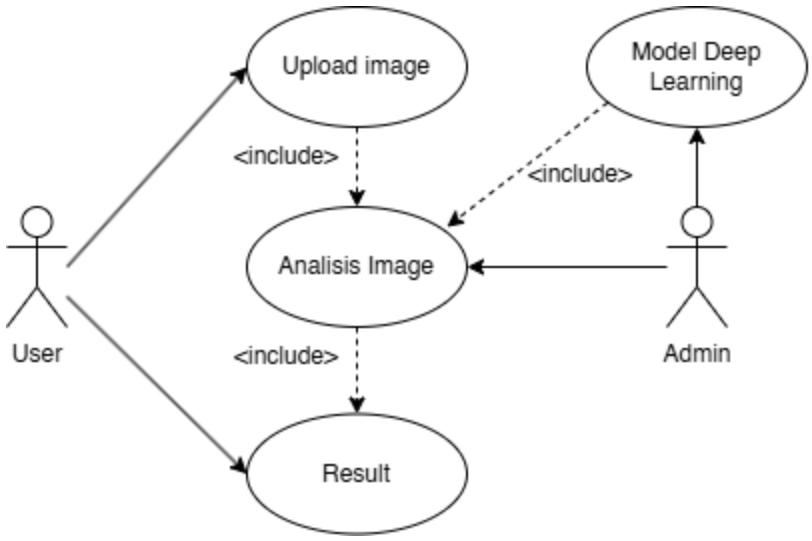


Gambar 2.3 *Flowchart Diagram*

Flowchart pada Gambar 2.3 menggambarkan langkah-langkah yang dilakukan dalam menggunakan aplikasi "Skinalyze". Proses dimulai dengan membuka laman web pada **https://tktb.sikester.my.id/**. Kemudian, user mengunggah image kondisi kulitnya dari *device* yang digunakan. Selanjutnya, sistem akan memverifikasi format file yang diunggah oleh *user*. Jika format file tidak sesuai maka user akan diarahkan untuk kembali mengunggah *image* yang sesuai. Ketika format file sudah sesuai, image akan dianalisis oleh model *deep learning*. Kemudian, hasil analisis menggunakan *deep learning* akan ditampilkan kepada *user*.

c. ***Use Case Diagram***

Pada bagian ini menjelaskan tentang Use Case Diagram yang menggambarkan interaksi antara pengguna dan sistem, serta antara admin dan sistem. Diagram ini memberikan gambaran yang jelas mengenai cara kerja sistem dan menjelaskan berbagai tindakan yang dapat dilakukan oleh masing-masing aktor dalam sistem.



Gambar 2.4 *Use Case Diagram*

Gambar 2.4 diatas menjelaskan tentang interaksi antara aktor dan *use case* yang menjelaskan fungsionalitas utama yang dapat diakses oleh aktor yang terlibat di dalam sistem. Berikut dua aktor utama yang terlibat dalam sistem.

1. *User* / Pengguna

   User atau pengguna adalah pengguna yang mengakses laman web tanpa harus melakukan registrasi terlebih dahulu. Pengguna dapat mengakses fitur yang telah disediakan aplikasi ‘Skinalyze’ :

   - Fitur *Analyze* : Mengunggah image kulit untuk kemudian dianalisis sehingga didapatkan kondisi kulit sesuai dengan prediksi. Terdapat beberapa kondisi kulit yang dapat diprediksi, yaitu *Acne/Rosacea, Atopic Dermatitis, Eczema, Psoriasis/Lichen Planus* dan *Urticaria Hives*.

2. Admin

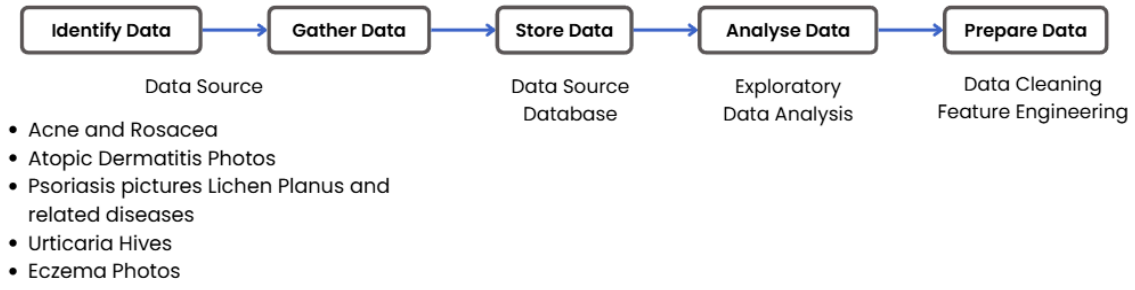
   Admin adalah individu yang memiliki akses penuh terhadap sistem dan dapat melakukan *maintenance* terhadap model *deep learning* yang akan mempengaruhi hasil analisis *image*.

BAB III
PERANCANGAN DAN IMPLEMENTASI SISTEM

a. *Software Development Life Cycle* (SDLC)

Dalam pengembangan proyek Capstone ini, dipilih metode SDLC Agile sebagai metode yang akan diterapkan. Keputusan ini didasarkan pada perbedaan alur aktivitas model yang harus dipertimbangkan. Proyek Capstone tidak akan diselesaikan dalam satu proses langsung, melainkan melalui beberapa iterasi, mulai dari pembuatan prototipe hingga penambahan fitur secara bertahap. Dengan alasan ini, metode SDLC Agile memberikan fleksibilitas yang lebih tinggi bagi pengembang dibandingkan dengan metode SDLC Waterfall. Model Waterfall dianggap tidak cocok karena alur aktivitasnya yang kaku, dengan satu langkah yang harus diselesaikan sebelum melanjutkan ke langkah berikutnya, tanpa ada iterasi. Sebaliknya, metode SDLC Agile memiliki alur yang lebih fleksibel dan memungkinkan pengembangan yang iteratif dan evaluasi yang berkelanjutan hingga mencapai hasil pengembangan perangkat lunak yang diinginkan.

b. Dataset



Gambar 3.1 Data Processing

1. Identify data
    - Identifikasi jenis data yang relevan untuk deteksi penyakit kulit
    - Sumber data yang digunakan meliputi gambar penyakit kulit seperti :
        a) Acne and Rosacea
        b) Atopic Dermatitis Photos
        c) Psoriasis pictures Lichen Planus and related diseases

d) Urticaria Hives
e) Eczema Photos

2. Gather data
    - Dataset berisi gambar dengan format yang bervariasi dengan folder-label yang menunjukan klasifikasi penyakit kulit
    - Sebanyak lebih dari 22 kategori penyakit kulit tersedia dalam dataset ini. Akan tetapi pada pengembangan dasar aplikasi ini hanya akan menggunakan 5 kategori penyakit kulit
3. Store data
    - Penyimpanan dilakukan untuk mempermudah aksesibilitas dan menghindari kehilangan informasi saat proses pengolahan data
4. Analyse data
    - Dilakukan *Exploratory Data Analysis (EDA)* terhadap dataset untuk memahami karakteristiknya:
        a) Jumlah gambar dalam setiap kategori
        b) Variasi ukuran gambar
        c) Keberadaan data yang tidak sesuai
    - Hasil EDA menunjukkan distribusi data yang tidak merata sehingga diperlukan teknik augmentasi untuk mengatasi ketidakseimbangan data
5. Prepare data
    a) Data cleaning
        - Menghapus gambar dengan resolusi rendah
        - Membersihkan data dari *outliers* seperti gambar yang tidak relevan atau salah klasifikasi
    b) Feature Engineering
        - *Resize* gambar ke dimensi seragam agar compatible dengan algoritma yang akan digunakan
        - Melakukan augmentasi data

c) Train-test split

- Dataset dipecah menjadi tiga bagian training, validation dan test untuk memastikan model dapat diuji secara objektif

c. Model Deep Learning

Proyek ini menggunakan model *Deep Learning* yang memanfaatkan VGG19 sebagai arsitektur dasar melalui metodologi *transfer learning*. VGG19, sebuah *Convolutional Neural Network (CNN)* yang telah dilatih sebelumnya, dipilih karena keampuhannya yang telah terbukti dalam tugas-tugas kategorisasi gambar. Model VGG19 dimulai dengan menggunakan *weight* yang telah dilatih sebelumnya dari dataset ImageNet, dengan menghilangkan lapisan atasnya untuk memfasilitasi penggabungan lapisan khusus yang dirancang untuk klasifikasi khusus penyakit kulit.

Untuk mempertahankan informasi yang telah dilatih sebelumnya dari VGG19, semua lapisan dalam model dasar dibekukan, sehingga mencegah pembaruan apa pun pada bobotnya selama pelatihan. Lapisan khusus tambahan dimasukkan di atas keluaran VGG19:

1. Lapisan *Flatten* untuk mengubah output dari lapisan konvolusional menjadi vektor satu dimensi.
2. Lapisan *Dense* yang terdiri dari 256 neuron dengan aktivasi ReLU, yang menggabungkan parameter yang dapat dipelajari untuk menyesuaikan dengan masalah klasifikasi saat ini.
3. Lapisan *Dense* berikutnya dengan 128 neuron dengan aktivasi ReLU untuk meningkatkan representasi fitur.
4. Lapisan output *Dense* penutup yang mencakup jumlah neuron yang sesuai dengan jumlah kategori penyakit kulit (misalnya, 5), menggunakan fungsi aktivasi *softmax* untuk menghasilkan probabilitas untuk klasifikasi multi-kelas.

Model ini dibuat dengan pengoptimal Adam dengan tingkat pembelajaran 0,001 untuk menyesuaikan bobot secara efektif selama pelatihan. Fungsi kerugian cross-entropi kategorikal jarang digunakan karena sifat tugas klasifikasi multi-kelas, dan akurasi dipilih sebagai metrik evaluasi.

Model ini menjalani pelatihan selama 25 epoch dengan ukuran batch 32, memanfaatkan dataset pelatihan dan memverifikasi terhadap dataset validasi yang berbeda. Pra Pemrosesan data memerlukan penskalaan gambar input menjadi 150 x 150 piksel agar sesuai dengan spesifikasi input VGG19 dan menstandarkan nilai piksel untuk meningkatkan stabilitas pelatihan.

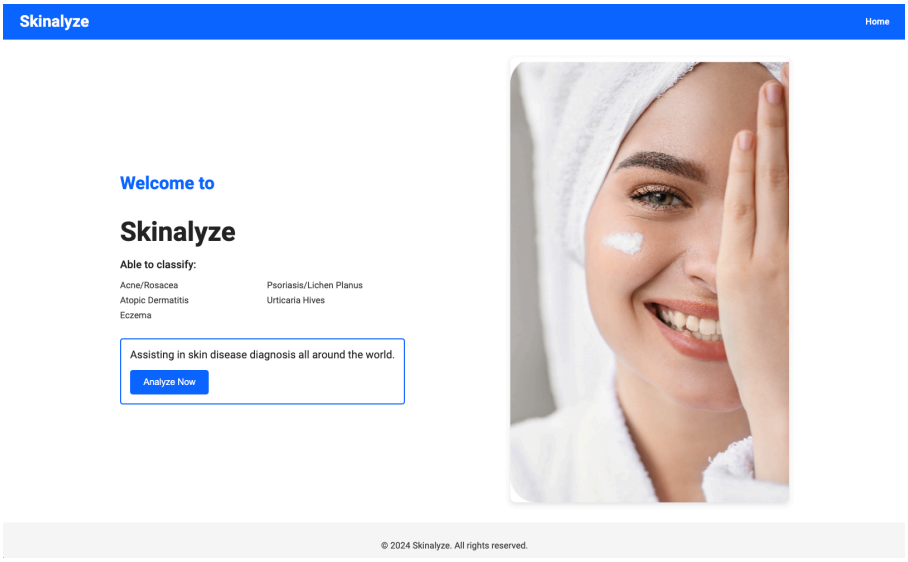
Arsitektur ini memanfaatkan kemampuan ekstraksi fitur yang kuat dari VGG19 sambil menyesuaikan dengan persyaratan kategorisasi spesifik aplikasi. Model akhir yang dilatih diekspor dalam format ONNX, sehingga memudahkan penerapan yang efisien di berbagai platform dan skenario.

d. Pengembangan Antarmuka

Tampilan layar dirancang dengan ukuran standard desktop (1700px x 1100px), dengan warna putih dan biru yang berkesan bersih dan terpercaya. HEX Code utama dari tiap laman adalah ##0C65FF, warna biru tua yang pekat. Mempermudah pengalaman pengguna, semua tulisan yang terdapat pada laman menggunakan penjajaran rata kiri dan rata tengah.

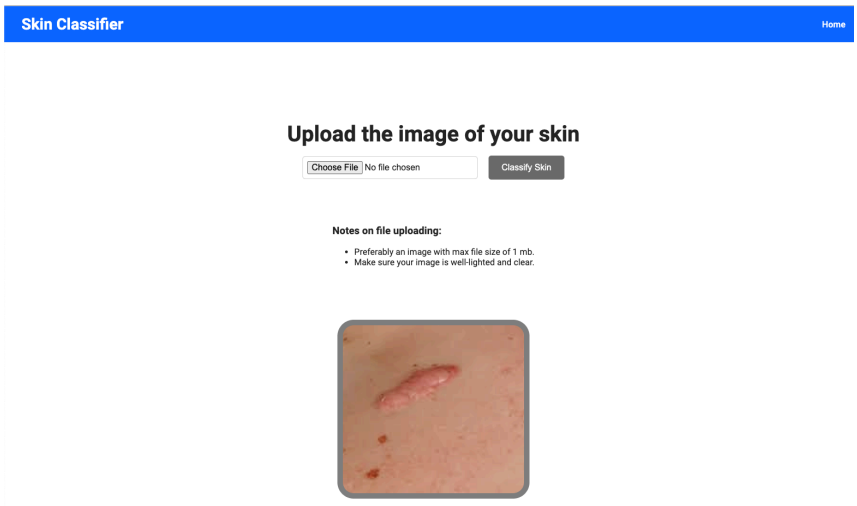
Tampilan layar yang dirancang pada *prototype* terdiri dari:

1. *Landing page*
2. *Image Input Page*
3. *Loader*
4. *Results*



Gambar 3.2. Landing Page Skinalyze

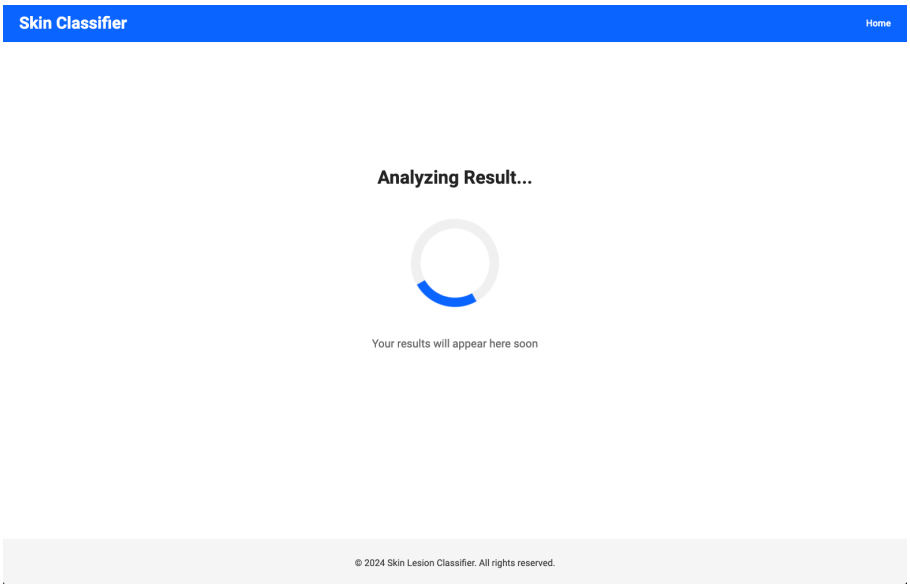
Landing Page berisi pengenalan dari jenis-jenis *skin lesion* yang bisa dianalisis, seperti rosacea, eksim, dermatitis atopik, psoriasis, dan *urticaria hives*. Terdapat tombol "*Analyze Now*" yang akan mengarahkan pengguna ke halaman untuk mengunggah citra.



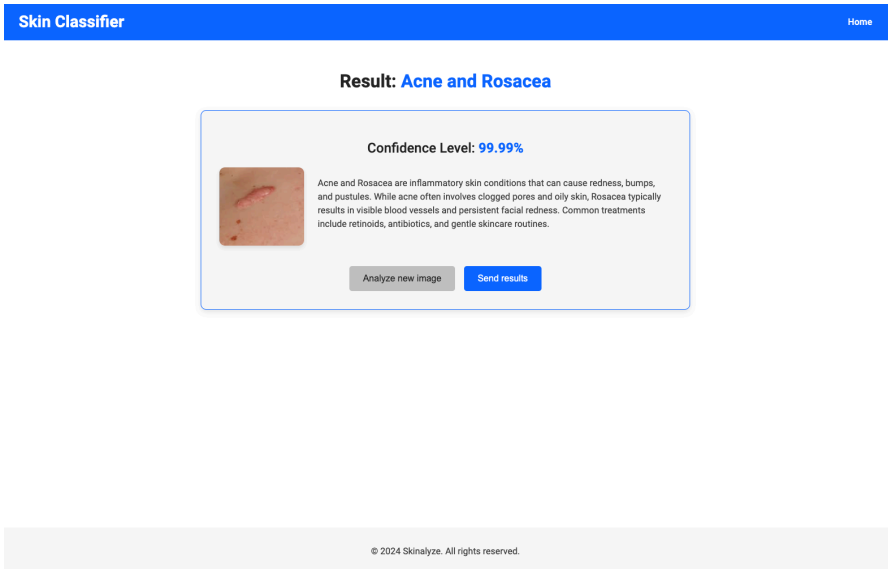
Gambar 3.3. Image Input Page

Pada laman ini, citra akan diunggah untuk selanjutnya dianalisis. Disertakan juga contoh citra yang baik untuk diunggah guna analisis, dengan keterangan kriteria yang jelas teruntuk besar ukuran *file* dan juga penerangan pada citra agar dapat dianalisis dengan lebih akurat.

Setelah mengunggah citra, pengguna dapat menekan tombol "*classify skin*" untuk memulai analisis. Saat citra dikelola, *loader* akan muncul sebagai *visual cue* untuk pengguna bahwa citra berhasil diterima sistem untuk dianalisis.



Gambar 3.4. Loader Page



Gambar 3.5. Results Page

Hasil akan tertampil pada laman dengan persentase akurasi/keyakinan dan deskripsi singkat dari hasil analisis. Laman ini dilengkapi dengan opsi untuk menganalisis citra lain, atau untuk melanjutkan dengan mengirim hasil.

e. *Techstack Frontend*

Front-end adalah bagian dari situs web yang dapat dilihat dan diinteraksikan oleh pengguna. Front-end mencakup elemen-elemen visual seperti graphical user interface (GUI), command line interface, desain, navigasi menu, teks, gambar, video, dan sebagainya [3]. Pada proyek ini digunakan HTML, CSS, dan Javascript untuk membuat bagian *frontend* dari sistem

1. HTML (HyperText Markup Language) merupakan bahasa *markup* standar yang digunakan untuk membuat suatu *website*. Bahasa ini memungkinkan pembuatan dan struktur bagian, paragraf, dan tautan menggunakan elemen HTML seperti tag dan atribut. HTML terdiri atas 3 komponen utama yaitu *opening tag*, content, dan *closing tag* [2]. HTML dipilih karena mudah digunakan untuk pemula dan dapat diintegrasikan dengan CSS untuk *styling* serta Javascript untuk fungsionalitas.

2. CSS (Cascading Style Sheets) merupakan bahasa pemrograman yang digunakan untuk menata elemen yang ditulis dalam bahasa *markup* seperti HTML Hubungan antara HTML dan CSS sangat erat karena HTML adalah fondasi utama dari sebuah situs dan CSS adalah semua estetika dari keseluruhan situs web [3]. Penggunaan CSS membuat tampilan website yang telah dibuat menggunakan HTML menjadi lebih menarik dan dinamis.

3. Javascript merupakan bahasa pemrograman yang dapat diintegrasikan dengan HTML dan CSS yang digunakan untuk memberikan interaktivitas pada halaman web.

Kombinasi penggunaan HTML, CSS, dan JavaScript ini memungkinkan pembuatan halaman web yang tidak hanya fungsional tetapi juga memiliki tampilan yang menarik dan pengalaman pengguna yang optimal.

f. *Techstack Backend*

Back-end adalah bagian dari situs web yang berfungsi di sisi server. Back-end menyimpan dan mengatur data serta memastikan segala sesuatu di sisi klien dari situs

web berfungsi dengan baik. Back-end tidak dapat dilihat atau diinteraksikan oleh pengguna secara langsung. back-end mencakup aktivitas seperti membuat API (Application Programming Interface), membuat library, dan bekerja dengan system component tanpa user interface (UI), serta sistem pemrograman ilmiah. [3]

Pada pengembangan teknologi backend dengan memperhatikan kebutuhan spesifik proyek, pengalaman tim pengembang, kebutuhan performa dan skalabilitas, serta ekosistem dan tools yang tersedia, FastAPI merupakan pilihan yang terbaik karena menawarkan kecepatan, kemudahan penggunaan, dan fitur-fitur modern yang sangat membantu dalam pengembangan API. Framework ini memungkinkan pembuatan API yang cepat dan efisien, dilengkapi dengan dokumentasi otomatis yang memudahkan tim pengembang dan pengguna memahami cara kerja API.

FastAPI juga mendukung pemrosesan data yang cepat, sehingga cocok untuk aplikasi yang harus menangani banyak permintaan sekaligus tanpa membuat sistem lambat. Selain itu, penggunaannya cukup sederhana dan tidak memerlukan banyak langkah rumit, sehingga memudahkan tim pengembang, termasuk mereka yang baru mulai belajar. Dengan kelebihannya ini, FastAPI menjadi pilihan yang sangat baik untuk proyek yang memerlukan API yang cepat, andal, dan mudah dikelola.

g. Kontainerisasi

Kontainerisasi adalah bagian penting dari arsitektur sebuah proyek, menggunakan Docker untuk menyediakan sebuah *environment* yang konsisten dan terisolasi untuk tahap pengembangan maupun produksi. Dengan melakukan kontainerisasi terhadap aplikasi, tim telah memastikan bahwa perangkat lunak akan memiliki sifat yang identik di mana pun perangkat lunak itu digunakan, hal ini mengeliminasi isu yang diakibatkan oleh perbedaan *environment.* Konsistensi ini adalah hal yang sangat penting untuk menjaga interaksi yang andal antara komponen *frontend* dan *backend.*

Dua buah kontainer dikembangkan sebagai bagian dari *deployment:*

1. **Kontainer *frontend*** menyajikan file statis, seperti HTML, CSS, dan JavaScript, memungkinkan pengguna untuk mengakses aplikasi melalui browser. Kontainer ini bertindak sebagai bagian yang berhadapan dengan pengguna dari sistem,

memastikan interaksi yang lancar dan respons yang cepat terhadap input pengguna.

2. **Kontainer *backend*** bertanggung jawab untuk mengelola logika aplikasi, khususnya menangani permintaan pengguna, melakukan pra pemrosesan gambar yang diunggah, dan menjalankan kesimpulan menggunakan model deep learning. Kontainer ini terintegrasi dengan **ONNX Runtime** untuk menjalankan prediksi secara efisien.

Dengan menggunakan Docker Compose, kontainer-kontainer ini dapat diatur untuk bekerja bersama dengan mulus, menyederhanakan proses penyiapan untuk lingkungan pengembangan, pengujian, dan produksi lokal. Pendekatan kontainerisasi juga menyederhanakan penskalaan aplikasi karena instance tambahan dari setiap kontainer dapat digunakan sesuai kebutuhan untuk menangani permintaan pengguna yang meningkat.

h. Deployment

Deployment memanfaatkan **Azure Virtual Machines (VM),** solusi komputasi awan yang memberikan fleksibilitas dan sumber daya yang dibutuhkan untuk infrastuktur yang kuat. Setiap VM diberi alamat IP publik, sehingga memungkinkan aplikasi diakses secara global. Penerapan ini termasuk menyiapkan **Nginx**, server web berkinerja tinggi dan *reverse proxy*, untuk mengelola dan merutekan lalu lintas yang masuk secara efisien. Nginx bertindak seperti gateway, mengarahkan lalu lintas berdasarkan aturan yang sudah diterapkan sebelumnya:

- Permintaan ke **tktb.sikester.my.id** dialihkan ke **kontainer frontend**, memastikan pengguna dapat mengakses antarmuka untuk mengunggah gambar dan melihat hasil.
- Permintaan ke **tktb-be.sikester.my.id** dialihkan ke **kontainer backend**, di mana titik akhir API memproses data dan mengembalikan hasil.

Penggunaan subdomain untuk frontend dan backend tidak hanya meningkatkan kejelasan dan organisasi tetapi juga meningkatkan keamanan dengan mengisolasi fungsi. Pemisahan ini memastikan bahwa setiap bagian dari aplikasi dapat dipelihara, dipantau, dan ditingkatkan secara independen.

Penerapannya dioptimalkan untuk **skalabilitas**, memungkinkan sistem untuk menangani beberapa permintaan bersamaan tanpa penurunan kinerja. Infrastruktur cloud Azure memberikan fleksibilitas untuk menskalakan sumber daya secara dinamis, memastikan aplikasi dapat beradaptasi dengan beban pengguna yang berfluktuasi. Misalnya, jika lalu lintas pengguna meningkat secara signifikan, instance VM tambahan dapat disediakan untuk menangani permintaan, meminimalkan waktu henti dan mempertahankan waktu respons. Strategi penerapan juga mencakup alat pemantauan untuk melacak kinerja server, sehingga memungkinkan penyesuaian proaktif terhadap alokasi sumber daya.

BAB IV
PENGUJIAN DAN EVALUASI

Pengujian aplikasi adalah langkah penting untuk memastikan bahwa aplikasi berfungsi dengan baik, efisien, dan memberikan pengalaman yang memuaskan bagi pengguna. Dalam konteks CI/CD (Continuous Integration/Continuous Deployment), pengujian yang dilakukan mencakup unit test, integration test, acceptance test, dan scalability test.

Setiap jenis pengujian memiliki tujuan yang berbeda, tetapi saling melengkapi dalam memastikan kualitas aplikasi secara keseluruhan. Unit test dan integration test dijalankan otomatis dalam pipeline CI setiap kali ada perubahan kode untuk memastikan aplikasi tetap stabil dan bebas dari bug. Acceptance test dilakukan dalam tahap deployment untuk memvalidasi apakah aplikasi sesuai dengan kebutuhan pengguna akhir, sedangkan scalability test dilakukan untuk memastikan aplikasi dapat menangani beban yang lebih besar secara efisien saat diterapkan di lingkungan produksi. Dengan mengintegrasikan pengujian-pengujian ini ke dalam pipeline CI/CD, proses pengembangan dan penerapan aplikasi menjadi lebih cepat dan lebih dapat diandalkan.

a. Unit Testing

Unit test adalah tahap pengujian yang fokus pada pengujian bagian-bagian kecil dari aplikasi, yang disebut unit. Unit ini biasanya berupa fungsi, metode, atau modul individual yang tidak bergantung pada bagian lain dari aplikasi. Unit test dilakukan untuk memastikan bahwa setiap bagian kecil dari aplikasi bekerja sebagaimana mestinya ketika diuji secara terpisah. Tujuannya adalah memverifikasi bahwa setiap unit aplikasi berfungsi dengan benar sesuai dengan spesifikasi, mendeteksi dan memperbaiki bug atau kesalahan pada tahap awal pengembangan, serta memastikan bahwa setiap unit berperilaku konsisten meskipun terdapat sebuah perubahan pada bagian kode lainnya. Pada proyek ini, pengujian unit dilakukan pada tiga komponen utama:

1) Komponen Machine Learning

   Menguji apakah model deep learning dapat menerima input gambar, memprosesnya, dan menghasilkan prediksi yang sesuai dengan akurasi yang baik.

Tes ini memastikan bahwa model berjalan dengan benar dalam kondisi lokal tanpa tergantung pada bagian lain dari sistem.

2) *Frontend*

   Menguji bagian antarmuka pengguna (UI) untuk memastikan bahwa elemen-elemen visual, seperti tombol unggah gambar dan hasil klasifikasi, berfungsi dengan baik dan tampil sesuai desain yang diinginkan.

3) *Backend*

   Menguji apakah API backend dapat menerima gambar, mengirimkannya ke komponen ML, dan memberikan respons yang sesuai kembali ke frontend. Tes ini memastikan bahwa backend dapat menangani permintaan secara independen dan tidak bergantung pada bagian lain dari aplikasi.

Pengujian unit ini dilakukan dalam skala lokal pada masing-masing komponen, tanpa melibatkan penggunaan Docker atau pengaturan lingkungan terintegrasi lainnya. Hal ini memastikan bahwa setiap bagian aplikasi dapat berfungsi dengan baik secara mandiri sebelum dilakukan pengujian lebih lanjut dalam konteks sistem yang lebih besar.

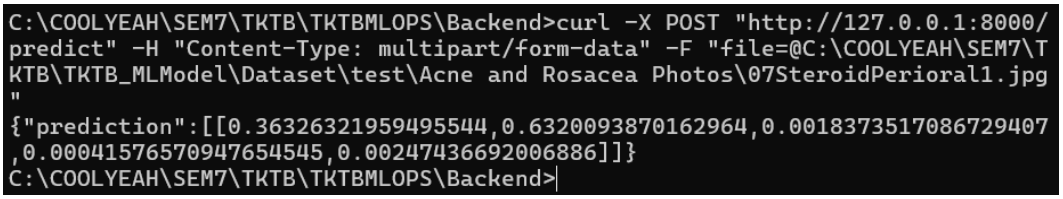
b. Integration Testing

Pengujian ini berfokus pada memastikan bahwa semua modul dalam aplikasi yang dikembangkan dapat bekerjasama dengan baik. Pengujian ini mencakup interaksi antara berbagai modul seperti autentikasi pengguna, pengelolaan data pasien, penyimpanan data, pengambilan data, serta pengiriman dan penerimaan data melalui API. Tujuannya adalah memastikan bahwa modul-modul aplikasi dapat berinteraksi dengan benar dan menghasilkan keluaran yang sesuai dengan ekspektasi. Mengidentifikasi dan memperbaiki masalah yang muncul akibat interaksi antar modul.Setelah pengujian unit, langkah selanjutnya adalah pengujian integrasi untuk memastikan bahwa berbagai komponen yang terpisah dapat bekerja sama dengan baik.

1) Pengujian API dengan Postman dan cURL

   Menggunakan Postman dan cURL untuk menguji integrasi antar API dari frontend ke backend, dan dari backend ke model ML. Ini termasuk pengujian untuk memastikan bahwa gambar yang diunggah dapat dikirim dengan benar ke

backend, diproses oleh model ML, dan hasil klasifikasi dapat dikembalikan ke frontend dengan tepat.



Gambar 4.1 Hasil Pengujian API dengan Postman dan cURL

2) Pengujian Error Handling

Menguji bagaimana sistem menangani kesalahan, seperti gambar yang tidak valid atau masalah koneksi antara komponen, serta memastikan bahwa sistem memberikan pesan yang informatif untuk membantu pengguna memahami masalah yang terjadi.

Pengujian ini memastikan bahwa komunikasi antar komponen bekerja dengan baik dan aplikasi dapat mengelola alur data secara efektif antar microservices.

c. Acceptance Testing

Acceptance test adalah pengujian yang dilakukan untuk menentukan apakah aplikasi memenuhi persyaratan dan harapan pengguna akhir. Pengujian ini biasanya melibatkan pengguna dari dunia nyata, seperti user atau dermatologist, yang akan menggunakan aplikasi dalam konteks operasional sehari-hari. Tujuannya yang pertama adalah untuk memastikan bahwa aplikasi sudah siap untuk digunakan dalam lingkungan produksi. Tujuan kedua adalah memvalidasi bahwa aplikasi memenuhi semua persyaratan fungsional dan non-fungsional yang telah ditentukan di awal proyek. Dan yang terakhir adalah memastikan bahwa aplikasi memenuhi persyaratan kepatuhan terhadap regulasi dan standar industri yang berlaku.

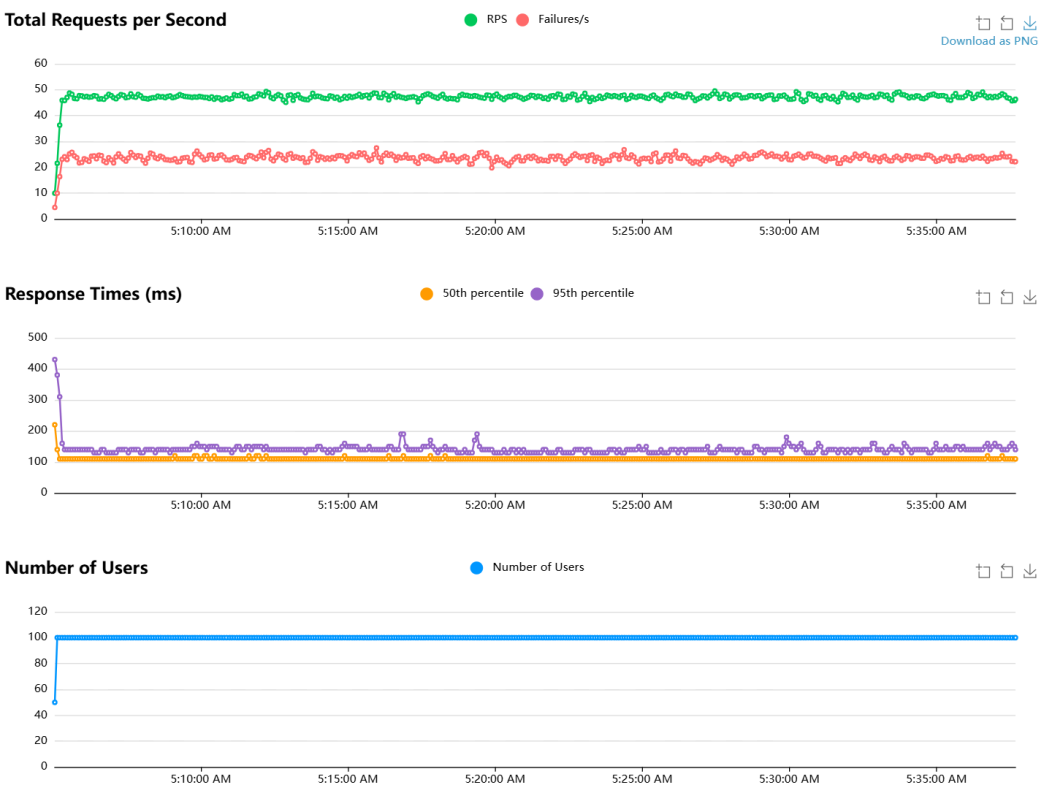
d. Scalability Testing

Pengujian skalabilitas bertujuan untuk mengevaluasi kinerja titik akhir API /predict di bawah beban hingga 100 pengguna secara bersamaan menggunakan Locust. Di samping itu, titik akhir root (/) diuji dengan permintaan GET sederhana untuk memberikan metrik kinerja dasar. Pengujian ini menggunakan permintaan POST ke

/predict dengan file gambar sampel (07SteroidPerioral.jpg, 94 KB) sebagai muatan. Meskipun mengirimkan 51.769 permintaan POST ke /predict, semuanya menghasilkan kesalahan **404 Not Found**, yang mengindikasikan adanya kesalahan konfigurasi backend atau titik akhir yang tidak terdefinisi. Sebaliknya, 51.764 permintaan GET ke / berhasil diproses tanpa kegagalan.

Metrik kinerja menunjukkan bahwa titik akhir root menunjukkan waktu respons rata-rata 130 milidetik, sedangkan titik akhir /predict mencatat rata-rata 150 milidetik, meskipun semua permintaan gagal. Waktu respons minimum dan maksimum berkisar antara 110 milidetik dan 170 milidetik untuk titik akhir root, dan antara 120 milidetik dan 190 milidetik untuk titik akhir /predict. Terlepas dari kesalahan yang terjadi, pengujian ini mencapai throughput agregat sebesar 169 permintaan per detik, dengan distribusi beban yang seimbang di kedua titik akhir. Kecepatan transfer data juga signifikan, mencapai 47,5 MB/detik di semua permintaan.

Pengujian ini menyoroti masalah kritis dengan titik akhir /predict, karena tingkat kegagalan 100% mengindikasikan bahwa backend tidak mengenali rute tersebut. Disarankan untuk memverifikasi konfigurasi aplikasi FastAPI, memastikan bahwa rute /predict didefinisikan dengan benar dan dapat diakses. Setelah menyelesaikan masalah ini, akan lebih baik untuk mengulangi uji skalabilitas untuk mengukur kinerja sebenarnya dari titik akhir. Selain itu, menerapkan mekanisme pencatatan dan pemantauan untuk backend dapat membantu mengidentifikasi dan menyelesaikan kesalahan konfigurasi seperti itu di masa depan. Meskipun sistem menunjukkan throughput dan waktu respons yang baik untuk titik akhir root, pengoptimalan lebih lanjut, seperti caching atau penyeimbangan beban, dapat meningkatkan latensi dan skalabilitas untuk lingkungan produksi. Untuk hasil lebih detail bisa dilihat pada Gambar 4.2.

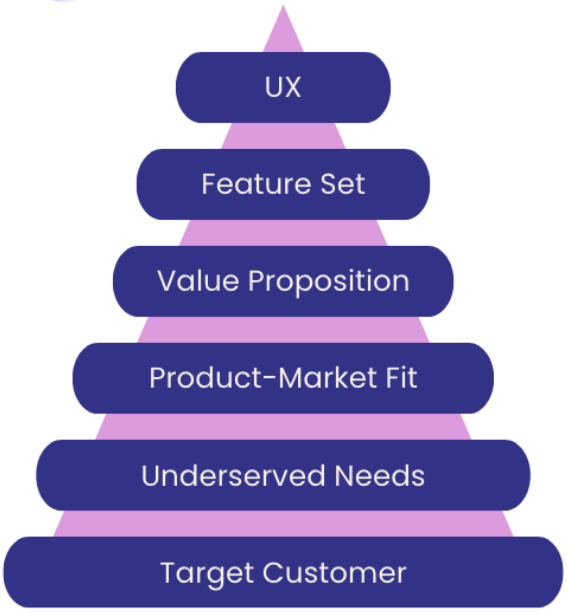


Gambar 4.2 Hasil Uji Skalabilitas

BAB V
ANALISIS SEGI BISNIS

Pasar dari pengembangan teknologi untuk *skin analysis* sangat besar secara global, mencapai USD123.8 juta di tahun ini dan terproyeksi untuk bertumbuh 10% tiap tahunnya, dengan pengguna terbanyak dari Amerika Utara, Eropa, dan Asia Pasifik (MarketDigits, 2024). Mengetahui potensi dari pengembangan teknologi ini menjadi sebuah bisnis, kami membuat analisis dan rencana penjualan (*business plan*) dalam bentuk piramida *market fit* dan juga *business model canvas*.

a. Product-Market Fit Pyramid



Gambar 5.1. Gambaran Piramida *Product-Market Fit*

Piramida ini adalah gambaran dari umpama rencana bisnis yang 2 unsur besarnya adalah pasar (*target customer, underserved needs*) dan produk (*value proposition, feature set,* dan UX).

Target Customer: yang menjadi pasar utama dari teknologi ini adalah para profesional di bidang dermatologi, seperti dokter kulit dan *aestheticians*. Hal ini didorong dengan alasan bahwa teknologi yang kami kembangkan tidak bisa menjadi pengganti diagnosis secara keseluruhan, namun untuk membantu diagnosis yang lebih cepat dan akurat.

Underserved Needs: karena kurangnya aksesibilitas ke ahli dermatologi yang murah, dan perawatan kulit di Indonesia masih terhitung keperluan tersier, masih sangat minim adanya perawatan kulit yang *personalized* menyesuaikan keperluan pasien, dimana produk yang tersedia hanya menyanggupi kategori yang luas seperti "kulit berminyak" dan "kulit kering" saja. Meski dengan berkembangnya *telemedicine*, masih banyak aspek yang dapat dikembangkan untuk *monitoring* kondisi kulit di kondisi *remote area*.

Value Proposition: Skinalyze sebagai solusi inovatif untuk mendeteksi penyakit kulit secara otomatis dengan menggunakan teknologi berbasis kecerdasan buatan.

Feature Set: fitur-fitur yang dapat diujian sebagai MVP (*Minimum Viable Product*) adalah pengunggah citra pada situs web yang kemudian diproses dengan Machine Learning yang terlebih dulu diolah dengan dataset.

UX: MVP akan diuji dalam bentuk prototipe dan juga *web testing*.

b. Business Model Canvas

Gambar 5.2. *Business Model Canvas* Skinalyze

Business Model Canvas adalah sebuah kerangka strategis yang digunakan untuk menyederhanakan penyampaian ide dan konsep suatu bisnis. Bagi Skinalyze, bisnis ini

bertujuan untuk mengatasi masalah akses terbatas ke dokter kulit, biaya konsultasi yang tinggi, serta kesulitan dalam mendapatkan diagnosis yang cepat dan akurat untuk penyakit kulit. Masalah ini sering menyebabkan keterlambatan dalam pengobatan karena tantangan dalam proses diagnostik. Sebagai alternatif yang ada, pasien biasanya mengandalkan janji temu tatap muka dengan dokter kulit, konsultasi telemedicine (jika tersedia), atau bahkan melakukan diagnosis mandiri melalui pencarian internet yang seringkali tidak akurat.

Solusi yang ditawarkan adalah aplikasi berbasis machine learning yang memungkinkan diagnosis penyakit kulit secara otomatis dan akurat. Aplikasi ini dapat diakses kapan saja melalui situs web dan dirancang untuk memberikan diagnosis cepat, akurat, dan terjangkau, dengan pengembangan berkelanjutan untuk menjaga kualitas dan relevansi. Solusi ini akan memberikan nilai lebih dengan analisis *skin lesion* instan yang didukung oleh teknologi machine learning yang sudah terlatih.

Pencapaian yang diukur meliputi tingkat akurasi diagnosis, jumlah pengguna aktif dan tingkat pertumbuhannya, waktu yang dibutuhkan untuk memberikan diagnosis, serta pengurangan waktu yang diperlukan untuk memulai pengobatan. Keunggulan kompetitif dari bisnis ini adalah kolaborasi dengan para ahli dermatologi untuk validasi dan penerapan berkelanjutan, serta keuntungan menjadi pelopor teknologi canggih yang memiliki nilai pasar tinggi secara global.

Saluran distribusi utama (*revenue stream*) mencakup akses langsung melalui aplikasi web, kemitraan dengan penyedia layanan kesehatan dan klinik kecantikan, serta pemasaran online seperti SEO, media sosial, dan konten pemasaran. Segmen pelanggan utama meliputi penyedia layanan kesehatan yang membutuhkan alat diagnosis jarak jauh yang efisien, serta individu yang mencari diagnosis cepat untuk masalah kulit mereka. Pengguna awal (*early adopters*) yang diincar adalah dermatologis, *aesthetician*, dan ahli perawatan kulit lain yang melek terhadap teknologi.

Struktur biaya mencakup pengeluaran untuk pengembangan, infrastruktur dan *hosting cloud*, biaya operasional, pemasaran dan akuisisi pelanggan, serta biaya kepatuhan dan regulasi. Sumber pendapatan utama dapat berasal dari biaya langganan untuk fitur premium atau akses tanpa batas, biaya per diagnosis, ataupun biaya lisensi dari penyedia layanan kesehatan yang menggunakan Skinalyze.

BAB VI
KESIMPULAN DAN SARAN

a. Kesimpulan

Proyek Skinalyze berhasil dikembangkan sebagai aplikasi berbasis teknologi deep learning yang dirancang untuk mempermudah deteksi penyakit kulit melalui klasifikasi otomatis gambar kulit. Dengan arsitektur microservices, aplikasi ini mampu mengintegrasikan tiga komponen utama (frontend, backend, dan model machine learning) untuk memberikan hasil diagnosis yang cepat dan akurat. Pengujian menyeluruh melalui unit test, integration test, acceptance test, dan scalability test memastikan bahwa setiap komponen aplikasi berjalan sesuai harapan, baik secara individu maupun terintegrasi. Aplikasi ini diharapkan dapat menjadi solusi yang efisien untuk meningkatkan aksesibilitas diagnosis awal penyakit kulit, terutama di daerah dengan keterbatasan akses tenaga medis.

b. Saran

Dari penelitian dan proses yang telah dilakukan, terdapat beberapa saran yang dapat diusulkan untuk penelitian berikutnya. Sorotan utama adalah pengembangan antarmuka pengguna (UI/UX) yang lebih *seamless* untuk memberikan pengalaman yang intuitif dan menarik bagi pengguna. Hal ini mencakup desain yang lebih responsif, tata letak yang lebih efisien, serta optimalisasi alur kerja agar pengguna dapat menggunakan aplikasi dengan mudah. Selain itu, perhatian lebih harus diberikan pada *ethical considerations,* seperti memastikan transparansi dalam proses prediksi, pengelolaan data yang mematuhi standar privasi dan keamanan, serta penyampaian hasil klasifikasi yang tidak menyebabkan misinformasi atau kebingungan di kalangan pengguna. Penelitian di masa depan juga dapat mencakup kolaborasi dengan tenaga medis untuk memastikan aplikasi memenuhi kebutuhan klinis dan etika medis yang berlaku.

LAMPIRAN

a. **Link GitHub**

https://github.com/giillbertt/TKTB_SkinDiseaseClassification

b. **Link WebApp**

https://tktb.sikester.my.id/

c. **Link Presentasi**

https://www.canva.com/design/DAGXj8YuWVk/Pj2qAoJqSnkiQoLxnygzOA/edit?utm_content=DAGXj8YuWVk&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

DAFTAR PUSTAKA

[1] Li, H., Pan, Y., Zhao, J., & Zhang, L. (2021). Skin disease diagnosis with deep learning: A review. *Neurocomputing*, *464*, 364-393.

[2] MarketDigits. (2024). Skin Analysis Technologies Market.

[3] G. (2023, April 18). Frontend vs Backend. GeeksforGeeks. https://www.geeksforgeeks.org/frontend-vs-backend/ [Accessed December 5th, 2024].

[4] Hostinger, "What is HTML? A Beginner's Guide to HTML," *Hostinger Tutorials*. https://www.hostinger.com/tutorials/what-is-html. [Accessed: December 9, 2024].

[5] Hostinger, "What is CSS? A Beginner's Guide to CSS," *Hostinger Tutorials*. https://www.hostinger.com/tutorials/what-is-css. [Accessed: December 9, 2024].

[6] Salim, Saja & Al-Tuwaijari, Jamal. (2021). Skin Disease Classification System Based on Machine Learning Technique: A Survey. IOP Conference Series: Materials Science and Engineering. 1076. 012045. 10.1088/1757-899X/1076/1/012045.

[7] Huyen, Chip. (2022). Designing Machine Learning Systems: An Iterative Process for Production-Ready Applications. O'Reilly Media, Inc.