Pengembangan Aplikasi Mobile Klasifikasi Penyakit Kulit Berbasis *EfficientNet-B0*, Arsitektur MVVM dan CI/CD *Pipeline*

Ichlasul Fikri Astamar Putra[1] dan Habibullah Akbar[2]

[1]Fakultas Ilmu Komputer, Program Studi Teknik Informatika, Universitas Esa Unggul
[2]Fakultas Ilmu Komputer, Program Studi Magister Ilmu Komputer, Universitas Esa Unggul
E-mail: ichlasul.ap@gmail.com, habibullah.akbar@esaunggul.ac.id

Abstrak

Penyakit kulit sering dianggap sebagai hal yang normal, tetapi dalam beberapa kasus, penyakit kulit dapat berbahaya dan mematikan dan seringkali dianggap abaikan oleh masyarakat luas. Disisi lain, saat ini teknologi berperan penting dalam kehidupan manusia sehari – hari sehingga aplikasi pada smartphone menjadi kebutuhan harian. Penelitian ini akan menjelasakan mengenai pengembangan aplikasi kesehatan kulit yang mengintegrasikan model machine learning dalam penggunaan aplikasi mobile berbasis Android menggunakan metode pengembangan *Extreme Programming* yang mengedepankan fleksibilitas dan responsif tergantung kebutuhan pengguna juga menekankan komunikasi yang erat antara tim pengembang. Selain itu penelitian ini juga berfokus dalam penerapan pada arsitektur aplikasi yang di rekomendasi oleh Android yaitu menggunakan *Model-View-ViewModel* (MVVM) dengan tingkat pengujian Black-Box Testing yang memuaskan dan nilai *System Usability Scale 92* menandakan aplikasi yang dibuat harapannya dapat diterima dan membantu masyarakat sebagai penanganan tahap awal atau para profesional kesehatan, termasuk dermatologis dalam memberikan perawatan yang lebih baik dan lebih tepat bagi pasien yang mengalami masalah kulit.

Kata kunci: *Aplikasi, Android, Pola Arsitektur, MVVM (Model-View-ViewModel), Kesehatan Kulit.*

Pendahuluan

Penyakit kulit termasuk dalam *International Statistical Classification of Diseases and Related Health Problems* (ICD) [1]. Meskipun tidak menular, penyakit kulit dapat menurunkan kualitas hidup penderitanya dan menimbulkan dampak sosial dan ekonomi bagi keluarga dan masyarakat. Berdasarkan Peraturan Menteri Kesehatan No. 71 Tahun 2015 tentang penanggulangan penyakit tidak menular, penyakit ini harus ditanggulangi dengan sebaik-baiknya. Masalah kesehatan kulit seperti dermatitis kontak memiliki prevalensi yang tinggi dan sering terjadi pada kelompok perempuan reproduktif (15-49 tahun) [2]. Jenis penyakit kulit lainnya yang juga sering ditemukan adalah kudis (scabies) [3], panu [4], dan urtikaria [5.

Berbagai upaya penanggulangan kesehatan kulit telah dilakukan oleh pemerintah maupun swasta. Contohnya adalah ketersediaannya poli kulit dan kelamin yang dapat ditemukan di rumah sakit ataupun fasilitas kesehatan lainnya serta berbagai program sosialiasi masyakarat. Selain itu, penderita penyakit kulit juga data berkonsultasi dengan dokter spesialis secara online dengan menggunakan seperti aplikasi Halodoc.

Secara akademik, beberapa peneliti di Indonesia juga telah mengembangkan berbagai aplikasi terkait kesehatan kulit. Penelitian [6] menggunakan metode *forward chaining* untuk membangun sistem pakar klasifikasi penyakit kulit. Jenis penyakit ditentukan berdasarkan aturan terhadap gejala-gejala yang muncul. Sayangnya, tidak ada pengujian yang dilakukan terkait metode yang digunakan. Pengujian biasanya dilakukan pada penelitian yang telah menggunakan metode yang lebih kompleks seperti *support vector machine* (SVM) [7], Naïve Bayes [8], c*onvolutional neural network* (CNN) [9], dan *a combination of MobileNet V2* and *Long short-term memory* (LSTM) [10]. Berdasarkan eksperimen yang dilakukan [9], CNN memiliki performa efektivitas yang lebih baik dibandingkan dengan SVM, *Naïve Bayes, Logistic Regression*, dan *Ran-*

DOI : http://dx.doi.org/10.32409/jikstik.23.4.3676

dom Forest.

Kebanyakan penelitian yang telah dibahas hanya berfokus pada satu aspek yaitu pada pengembangan metode klasifikasi secara otomatis. Belum banyak literatur yang membahas bagaimana proses implementasi model klasifikasi pada aplikasi dapat dilakukan dengan lebih sistematis. Sebenarnya, penelitian [11] telah mengusulkan proses konversi model deteksi penyakit kulit (berbasis CNN) menjadi format *TensorFlow Lite* (TFLite) yang kemudian dapat ditanamkan secara langsung pada bagian UI aplikasi Android (front-end). Cara ini dapat menghindari latensi jaringan, berfungsi secara offline, namun akan menyulitkan ketika model klasifikasi perlu diperbarui. Padahal, pembaruan aplikasi akan diperlukan saat terdapat model klasifikasi yang lebih baik ataupun adanya jenis penyakit kulit lain yang perlu dilatihkan. Selain itu, hal ini akan menyebabkan ukuran file aplikasi menjadi semakin besar sesuai kompleksitas model klasifikasi yang digunakan.

Penelitian ini bertujuan untuk mencari metode pembaruan aplikasi yang lebih baik dari pada yang sudah ada. Hal ini dilakukan dengan penerapan arsitektur *Model-View-ViewModel* (MVVM) dalam pengembangan aplikasi. Arsitektur ini berusaha mengatasi masalah pada arsitektur lain seperti *Model-View-Controller* (MVC) dimana Controller akan kesulitan ketika harus menangani banyak tugas. Hal yang serupa juga terjadi pada Presenter pada arsitektur MVP (*Model-View-Presenter*). Selain itu, MVC dan MVP hanya cocok aplikasi yang relatif kecil.

Dengan pemisahan antara logika bisnis yang mengatur pemrosesan data (Model), antarmuka (UI atau *user interface*) pengguna (*View*), dan *View-Model* yang menghubungkan antara *Model* dan *View*. Keunggulan pendekatan ini adalah perubahan pada data secara otomatis akan diperbarui di UI, dan sebaliknya, setiap perubahan interaksi pengguna di UI dapat langsung mempengaruhi data [5]. Untuk menangani perubahan pada aplikasi, kami akan menggunakan API yang diletakkan pada *ViewModel*. Dengan demikian, ketika model klasifikasi diubah ataupun jenis penyakit ditambah maka proses pembaruan aplikasi dapat ditangani dengan lebih efisien. Arsitektur MVVM sejalan dengan konsep Clean Architecture yang menekankan pemisahan bagian kode ke dalam beberapa lapisan berdasarkan tujuan masing-masing, sehingga bagian-bagian tersebut memiliki ketergantungan yang minimal.

Untuk memudahkan proses pembaruan, kami menggunakan pendekatan berbasis *DevOps* yaitu CI/CD [12][13], yang dapat menyatukan proses pengembangan, pengujian dan operasional. *Continuous Integration* bertugas untuk membantu pengembang mengintegrasikan perubahan kode secara lebih sering ke dalam cabang Bersama. Ketika perubahan dilakukan, pengujian otomatis diaktifkan untuk memastikan kode dapat digabungkan. *Continuous Delivery* (CD) kemudian mengotomatisasi rilis kode yang telah divalidasi ke repositori setelah proses build dan pengujian di CI selesai. Untuk menjalankan continuous delivery yang efektif, CI harus sudah ada dalam *pipeline* pengembangan. Dengan demikian, tim operasional dapat dengan cepat menerapkan aplikasi ke produksi. Pada penelitian ini, kami menggunakan platform *GitHub Actions* untuk menerapkan pendekatan CI/CD.

Metode Penelitian

Penelitian ini dilakukan dengan pendekatan yang diturunkan dari kombinasi *machine learning operations* (MLOps) [14] dan extreme programming. Terdapat 6 tahapan utama yang digunakan pada penelitian ini yaitu:

1. Inisisasi produk MLOps: kami melakukan analisis kebutuhan sesuai domain penyakit kulit menggunakan metode wawancara kepada dokter di klinik Estetik yang juga seorang konsultan hukum kesehatan. Selain itu, kami juga menggunakan kuesioner untuk mendapatkan kebutuhan target pengguna.

2. Tahapan rekayasa fitur: data input didapatkan dari citra kulit user yang akan diklasifikasi menggunakan model EfficientNet-B0 [15].

Tabel 1: Model Arsitektur EfficientNet-B0

Lapisan	Spesifikasi
Input layer	224x224x3
Base model	pooling: max
Batch normalization	Axis: -1 momentum: 0,99 epsilon: 0,001
Dense layer	Neuron: 256 kernel regularizer: L2 0,01 bias regularizer: L1 0,008 activation: ReLU
Dropout rate	0.3
Dense layer	neuron: 6 activation: SoftMax

Model ini memiliki keseimbangan yang cukup baik antara resolusi dan dimensi jaringannya (jumlah lapisan serta ukuran dan kedalam ekstraksi fitur). oleh karena itu EfficientNet-B0 digunakan oleh penulis. Arsitektur EfficientNet-B0 terdiri dari beberapa hyperparameter yang dapat dilihat pada Tabel 1.

3. Eksperimentasi: dilakukan untuk mencari performa (akurasi) model klasifikasi yaitu EfficientNet-B0 yang dapat diterima (setidaknya lebih dari 90%). Adapun jenis penyakit kulit terdiri dari dermatitis atopik, dermatitis kontak alergi, dermatitis seboroik, dan neurodermatitis.

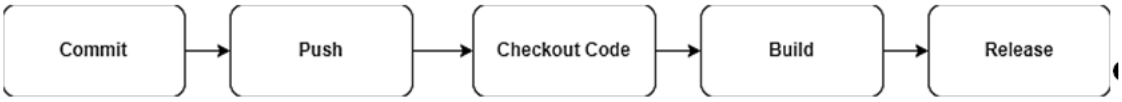
4. Desain aplikasi: aplikasi kesehatan kulit dirancang menggunakan arsitektur MVVM. Arsitektur MVVM memodelkan aplikasi kesehatan kulit menjadi 3 bagian. Bagian View mengatur bagian *front-end* berupa layout yang berinteraksi langsung dengan pengguna.

   Pada lapisan Model, terdapat 2 sumber data untuk aplikasi dimana penyimpanan lokal akan disimpan dan mengambil data pada SQLite kemudian untuk data remote akan disimpan dan mengambil data pada API, kedua data ini kemudian dijembatani dan diorganisir menggunakan pola repository untuk diakses dan juga dilakukan manipulasi data yang dihasilkan dari logic yang tersedia pada ViewModel, masing- masing ViewModel juga terdapat LiveData untuk diambil oleh lapisan View sehingga data yang ditampilkan pada View merupakan hasil binding data antar keduanya dan akan selalu di-update setiap kali ada perubahan data pada ViewModel.

   Pada bagian tengah, ViewModel bertugas untuk sinkronisasi antara View dan Model. Seperti yang dijelaskan sebelumnya, perubahan pada data secara otomatis akan diperbarui di UI, dan sebaliknya, interaksi pengguna di UI dapat langsung mempengaruhi data pada bagian back-end. Adapun pada bagian front-end aplikasi, kami menggunakan UML dan pemodelan UI menggunakan style guide untuk menjaga konsistensi.

5. Implementasi: penerapan dari hasil rancangan UI dilakukan menggunakan Bahasa Kotlin yang memiliki 100% interoperabilitas dengan Java, Android Studio, dan UI library tools Jetpack Compose. Adapun implementasi back-end dilakukan pada *Google Cloud Platform* (GCP) menggunakan layanan *Compute Engine*.

6. Tahap rilis dilakukan mencakup setup environment, building aplikasi mobile, dan deployment. Proses setup environment, building aplikasi, dan deployment dilakukan dengan pendekatan CI/CD pipeline menggunakan *GitHub Actions* yang mencakup *Commit, Push, Checkout Code, Build,* dan *Release* (lihat Gambar 1). Lingkungan pengembangan diatur secara otomatis sebelum proses build aplikasi *mobile* dan backend setiap kali ada perubahan kode. Kemudian proses model machine learning di-deploy secara otomatis kedalam lingkungan produksi (staging) dengan memanfaatkan *Docker Container* sebagai development environment untuk menjadikan aplikasi sebagai paket yang portable beserta dependensinya. Adapun, model EfficientNet-B0 di-deploy menggunakan kerangka OpenAPI. Selain itu, aplikasi usulan juga diuji menggunakan blackbox dan system usability scale (SUS).



Gambar 1: Diagram CI/CD Pipeline

Hasil dan Pembahasan

Hasil penelitian dijelaskan sesuai dengan tahapan MLOps. Pertama, hasil wawancara menunjukkan bahwa secara umum masyarakat belum memiliki kesadaran akan pentingnya kesehatan kulit. Ahli juga berpendapat bahwa kasus penyakit kulit semakin meningkat karena sifatnya menular. Hal ini bertentangan dengan Peraturan Menteri Kesehatan No. 71 Tahun 2015 yang menganggapnya sebagai penyakit yang tidak menular. Pengembangan aplikasi harus memiliki performa yang tinggi, dapat meningkatkan kesadaran Masyarakat umum serta menjaga privasi data.

Adapun dari kuesioner, kami berhasil mendapatkan jawaban dari 39 responden. Kebanyakan responden berusia antara 21-22 tahun, 71,8% adalah laki-laki (28,2% perempuan), dan kebanyakan adalah pelajar (mahasiswa). Hasil kuesioner menunjukkan bahwa 28,2 persen yang pernah mengalami kesehatan kulit (sisanya tidak pernah), 77% kesulitan untuk mendapatkan informasi kesehatan kulit, dan sebesar 48,7% pernah melakukan melakukan konsultasi/cek kesehatan kulit yang disebabkan oleh masalah seperti skabies, alergi, jerawat, ketombe, tungau, biang keringat, dan penggunaan salep yang tidak efektif. Sebesar 51,3% tidak pernah melakukan disebabkan berbagai alasan seperti takut biaya yang mahal, tidak menganggap penting, dan malas ke dokter.

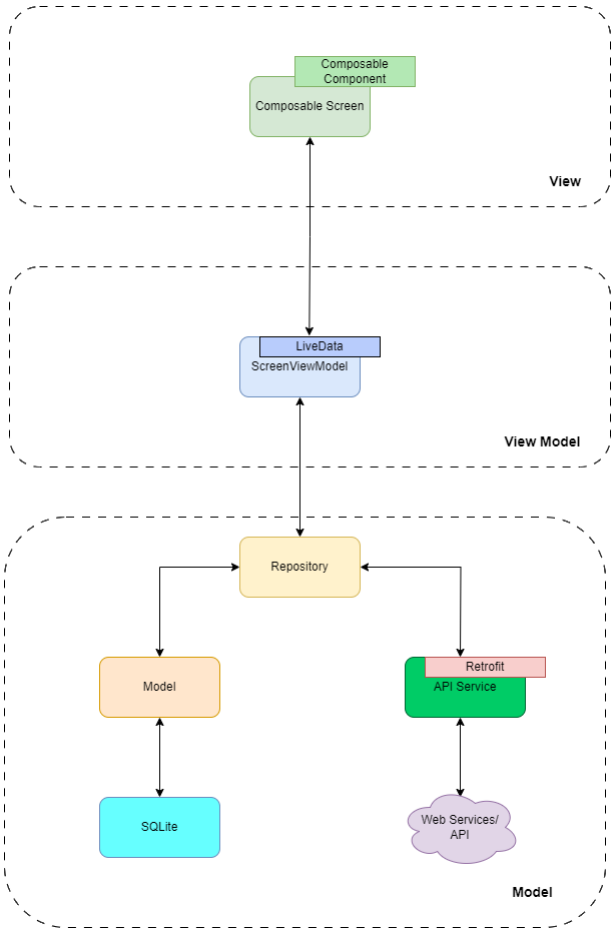
Selain itu, hanya 10.3% responden pernah memanfaatkan aplikasi (TroveSkin dan Halodoc) untuk konsultasi kesehatan kulit. Namun, sebesar 92,3% responden merasa kehadiran aplikasi sebenarnya dapat membantu dalam mengatasi masalah kesehatan kulit. Mereka menyarankan fitur yang berguna pada suatu aplikasi melingkupi fitur artikel kesehatan kulit, deteksi penyakit kulit otomatis dari

foto kulit, rekomendasi solusi masalah kulit, konsultasi dengan ahli, video call, chat, biaya konsultasi yang rendah. Deteksi penyakit dapat dilakukan menggunakan model machine learning.

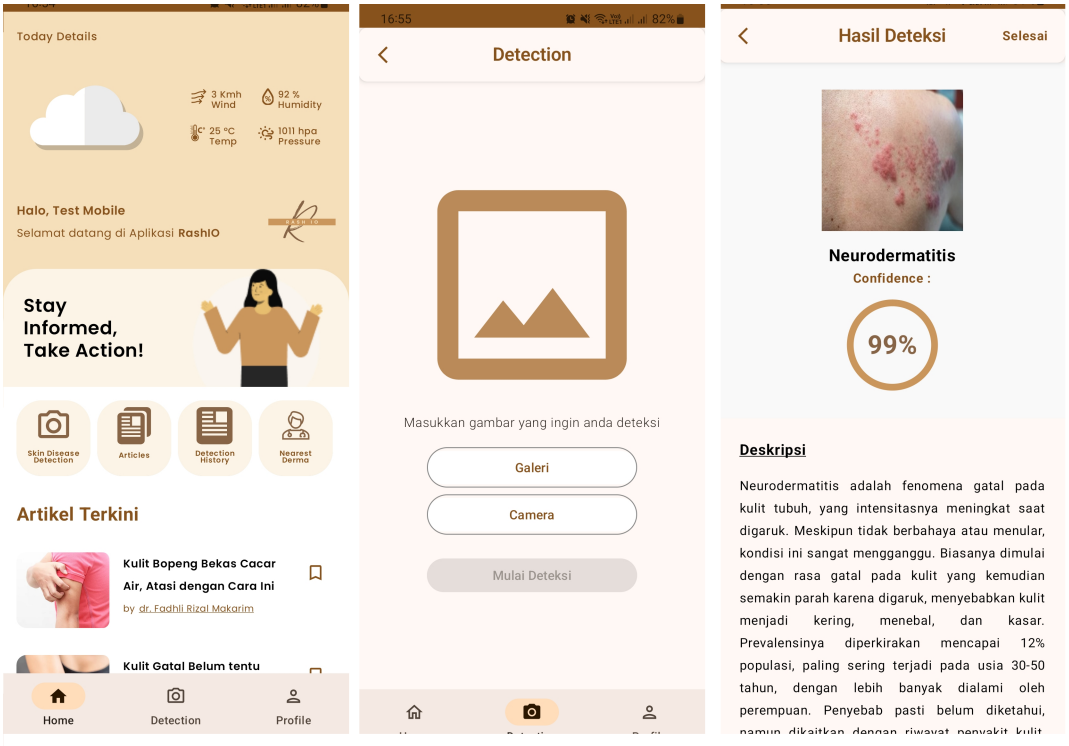
Berdasarkan hasil analisis kebutuhan tersebut dapat disimpulkan bahwa dibutuhkan adanya aplikasi kesehatan kulit yang dapat memenuhi berbagai kebutuhan user diatas. Data utama yang dibutuhkan pada aplikasi adalah berupa citra kulit yang dapat diproses menggunakan model klasifikasi CNN yaitu EfficientNet-B0. Berdasarkan hasil eksperimentasi, kami menemukan nilai akurasi EfficientNet-B0 dapat mencapai 93%. Nilai hyperparameter yang digunakan adalah learning rate sebesar 0.001, 30% dropout, iterasi training 25 kali (epoch), dan batchsize berukuran 64.

Blok diagram arsitektur MVVM yang digunakan dapat dilihat pada Gambar 2.

Aplikasi kesehatan kulit dapat menyimpan data secara lokal (SQLite) maupun eksternal (via API). Pada bagian View, terdapat 3 aktor yaitu pengguna, admin, dan pihak eksternal. Pengguna dapat melakukan pendaftaran dan autentikasi untuk melakukan identifikasi penyakit kulit (dengan mengunggah citra kulit), membaca artikel, mencari lokasi spesialis dermatologi terdekat, melihat riwayat identifikasi, dan mengubah profil. Admin dapat mengatur artikel kesehatan kulit dan rekomendasi solusi penyakit kulit. Pada setiap halaman aplikasi, terdapat komponen deklaratif yang dibuat dengan Jetpack Compose.



Gambar 2: Diagram MVVM Aplikasi Usulan



(a) (b) (c)

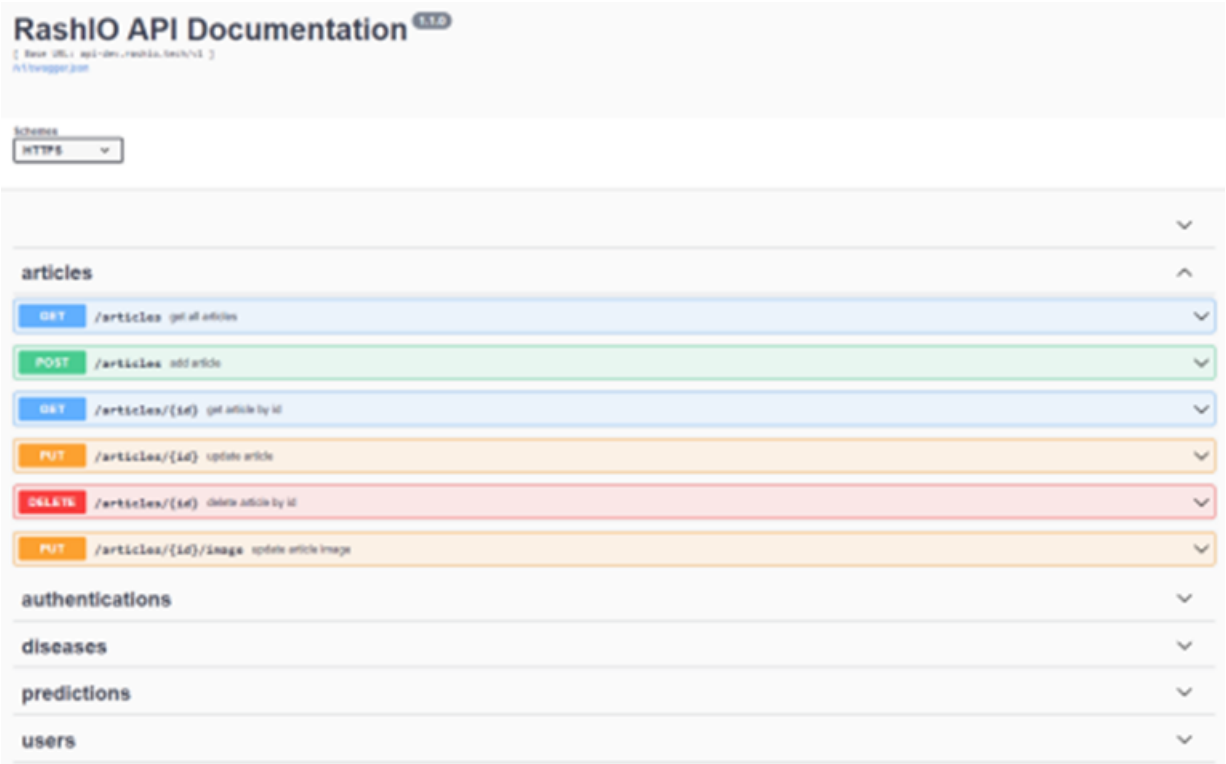
Gambar 3: Halaman Home (a) dan Tampilan Hasil Deteksi Penyakit Kulit (b) dan (c)

Halaman aplikasi yang telah dihasilkan mencakup Splash Screen, autentikasi, Home, artikel, deteksi penyakit kulit, profil. Halaman Home dan deteksi penyakit kulit ditunjukkan pada Gambar 3.

Setelah pengguna berhasil melakukan login, maka pengguna akan diarahkan ke halaman Home yang berisikan berbagai menu seperti deteksi penyakit kulit (skin disease detection), artikel, riwayat deteksi (detection history), dan kondisi cuaca. Pada halaman deteksi penyakit kulit pengguna dapat memilih citra ingin dideteksi (dapat melalui galeri pengguna ataupun menggunakan kamera). Setelah memilih tombol Mulai Deteksi maka maka aplikasi akan menampilkan informasi hasil deteksi penyakit.

Selanjutnya, Kami melakukan pengujian blackbox untuk menguji fungsionalitas dari aplikasi yang di kembangkan. Dimana Dari Hasil pengujian blackbox menunjukkan bahwa seluruh skenario pengujian yang berjumlah 25 skenario pada halaman-halaman aplikasi dapat berfungsi sesuai harapan yang berarti aplikasi yang dibangun sesuai dengan ekspektasi. Selain itu, kami juga melakukan pengujian Usability Testing yang menggunakan System Usability Scale (SUS) dimana SUS umumnya diakui sebagai "standar industri" di sektor bisnis dan teknologi. Dari 15 responden, didapatkan skor sebesar 92 yang menunjukkan aplikasi ini dapat diterima oleh pengguna.

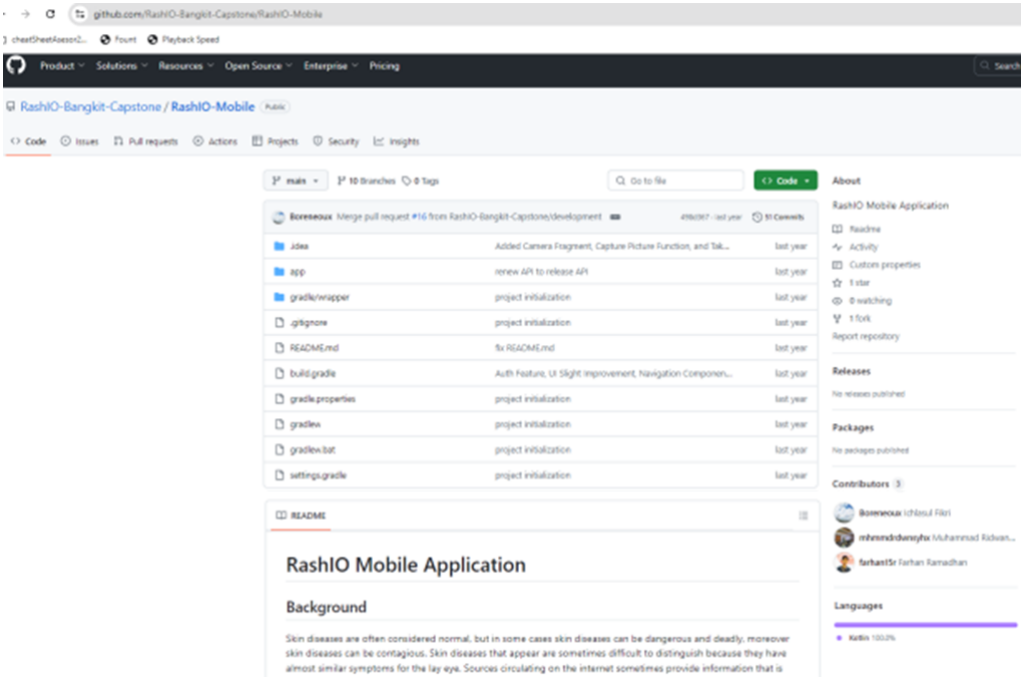
Bagian Model dapat mengakses sumberdaya eksternal di GCP menggunakan API. Terdapat 5 API yang digunakan pada aplikasi usulan yaitu UserAPI (untuk membuat akun bagi pengguna), AuthAPI (untuk validasi identitas pengguna), ArtikelAPI (untuk mengambil data artikel), PredictionAPI yaitu model machine learning yaitu EfficientNet-B0 (untuk identifikasi penyakit kulit), dan DiseaseAPI. Data pada Model diatur dalam package data dan domain, yang kemudian diproses oleh ViewModel. Keempat artikel tersedia pada endpoint API https://api-dev.rashio.tech/v1/documentation (lihat gambar 4).



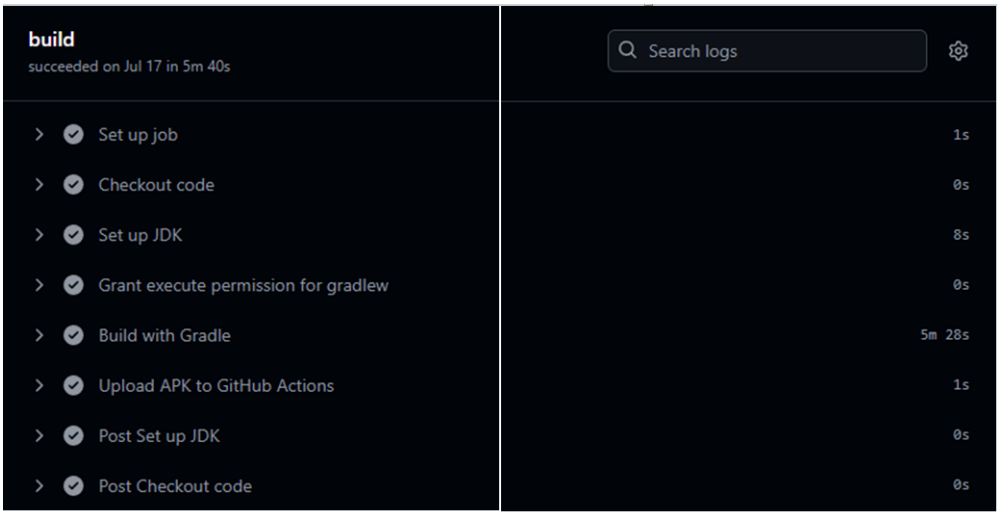
Gambar 4: Dokumentasi API Aplikasi Usulan

Dalam penulisan kode, kami menggunakan sistem back-end terpisah untuk database dan data lainnya, dengan memanfaatkan API untuk berkomunikasi antara aplikasi dan sistem back-end. Sistem back-end ini telah di-deploy pada GCP dengan menggunakan layanan Compute Engine. Selama proses deployment, Docker Container digunakan sebagai development environment yang terisolasi untuk aplikasi yang akan di-deploy. Hal ini memastikan bahwa proses deployment menjadi lebih mudah dan konsisten, dengan menyediakan paket aplikasi yang portable dan menyelaraskan semua dependensi yang diperlukan. Selain itu, aplikasi juga menyediakan form syarat dan ketentuan dan kebijakan privasi data pada aplikasi untuk mengatasi kekhawatiran ahli mengenai consent mengenai regulasi privasi dan keamanan data pengguna yang dijadikan persyaratan pengguna ketika ingin melakukan registrasi akun. Selain itu, kami juga menggunakan Open-Meteo sebuah API cuaca yang akan digunakan pada halaman home untuk memberikan informasi cuaca. Ada-

pun ViewModel bertugas untuk menyediakan data yang dibutuhkan oleh View dimana setiap screen memiliki ViewModel yang terletak dalam package screen masing-masing. Pada ViewModel terdapat LiveData yang berelasi dengan View sehingga data pada View memiliki binding dan selalu update terhadap perubahan data. Dengan arsitektur seperti ini, diharapkan pekerjaan tim pengembang akan lebih terkelola karena setiap modul memiliki kebergantungan yang rendah. Selain itu, unit testing dapat dilakukan dengan lebih sistematis. Library yang digunakan untuk aplikasi ini adalah Room (untuk lapisan abstraksi untuk mengakses data SQLite), Retrofit (untuk berkomunikasi dengan data secara remote via API dalam bentuk HTTP Request), Hilt (sebagai dependency injection), Navigation Compose (untuk mengelola navigasi pada aplikasi Android), Coil (untuk mengelola citra kulit yang dimuat API), Location Service (untuk mendapatkan data lokasi pengguna dengan menggunakan GPS), Accompanist (untuk mengelola Android runtime permissions pada Jetpack Compose), dan Lottie (untuk melakukan render animasi berbasis JSON pada aplikasi Android). Kode sumber aplikasi usulan dapat diakses pada pada situs https://github.com/RashIO-Bangkit-Capstone/RashIO-Mobile (lihat gambar 5).



Gambar 5: Repositori Aplikasi Mobile Kesehatan Kulit Usulan



Gambar 6: Workflow CICD pipeline untuk Build Job

Implementasikan CI/CD pipeline kedalam repositori dilakukan dengan menambahkan worfklow pada repository, yang berisi tahapan dari Set up job, Checkout code hingga Complete job (lihat gambar 6). Pada tahapan ini APK aplikasi dibuat dan juga di-upload pada Github Actions.

Setelah membuat workflow yang berisikan job untuk build aplikasi, selanjutnya penulis melakukan uji coba terhadap job yang sudah dibuat. Seperti yang dapat dilihat pada Tabel 2, hasil testing job build pada Github Action menunjukkan proses mengalami kegagalan pada percobaan build 1-6. Hal ini disebabkan terdapat kesalahan versi JDK yang berbeda dengan JDK aplikasi yang dibuat pada penulisan workflow sehingga terjadilah kegagalan. Namun setelah mengubah versi JDK yang sesuai dengan versi (Java version 17) yang digunakan aplikasi maka pada build 7 dan setelahnya berhasil dilakukan. Waktu yang dibutuhkan untuk melakukan berbagai jenis pekerjaan, seperti perubahan kode dan update versi JDK, berkisar antara 5-7 menit.

Tabel 2: Hasil Job pada Github Action

Build	Github *Action Workflow*		
	Deskripsi	**Waktu (*minute* dan *second*)**	**Status**
1	*Config Build Job*	*Failure*	Gagal
2	*Config Build Job*	*Failure*	Gagal
3	*Config Build Job*	48s	Gagal
4	Perubahan Kode *Toggle Button Detection Screen*	20s	Gagal
5	*Config Build Job*	35s	Gagal
6	*Config Build Job*	17s	Gagal
7	*Config Build Job* (*Changed* JDK *version*)	6m 25s	Berhasil
8	*Config Build Job* (*Changed* JDK *version*)	6m 6s	Berhasil
9	*Config Build Job* (*Changed* JDK *version*)	6m 20s	Berhasil
10	Perubahan Kode *Top Bar* Detection *Result Screen*	6m 55s	Berhasil
11	Perubahan Kode List Artikel *Home Screen*	5m 56s	Berhasil
12	Perubahan Kode List Artikel *Home Screen*	5m 49s	Berhasil

Hasil-hasil diatas menunjukkan bahwa implementasi CI/CD memberikan keuntungan pengembangan aplikasi dimana proses membangun, menguji, dan mendistribusikan aplikasi dapat dilakukan dengan lebih otomatis, khususnya dalam menangani perubahan pada aplikasi. Hal ini sangat membantu dalam penambahan berbagai fitur aplikasi seperti dalam penggunaan API model deep learning seperti EfficientNet-B0 dan update berbagai kebutuhan aplikasi lainnya.

Penutup

Pengembangan aplikasi *mobile* klasifikasi penyakit kulit berdasarkan implementasi CI/CD pipeline telah dilakukan pada penelitian ini. Hasil penelitian menunjukkan penggunakan pola arsitektur *Model-View-ViewModel* (MVVM) dan metode pengembangan *Extreme Programming* dapat mengakomodasi modularisasi dan update aplikasi dengan lebih mulus. *Model deep learning EfficientNet-B0* dan perubahan seperti versi JDK dapat dilakukan pada aplikasi dengan lebih otomatis menggunakan *Github Action*. Peneliti melakukan tiga bentuk pengujian terhadap penelitian ini, yaitu *Black-Box Testing*, *System Usability Scale (*SUS), dan uji B*uild Job*. Hasil uji *Black-Box* menunjukkan aplikasi telah dibangun sesuai dengan analisis kebutuhan. Skor SUS (92) menunjukkan aplikasi dapat diterima oleh calon pengguna. Terakhir hasil uji *Build Job* membantu pengembang untuk melacak jika terjadi kesalahan pada proses CI/CD.

Ucapan Terimakasih

Ucapan terima kasih kami berikan kepada tim Rashio Capstone Project Bangkit 2023.

Daftar Pustaka

[1] Anonim , “Peraturan Menteri Kesehatan Republik Indonesia Nomor 71 Tahun 2015 Tentang Penanggulangan Penyakit Tidak Menular”, Permenkes, Jakarta Kemenkes RI, 2015.

[2] S. Suharno dan Y. Nugraha, “Pengetahuan Pasien tentang Perawatan Luka Dermatitis Kontak pada Pasien Rawat Jalan Berhubungan dengan Kejadian Dermatitis Infeksiosa”, J. Keperawatan Silampari, vol. 6, no. 2, pp. 980–986, 2023.

[3] F. L. Santoso, A. I. Anwar, F. Tabri, S. Amin, A. Adriani, I. Idrus, dan A. A Arsyad, “Profil Penyakit Kulit Penduduk Perkampungan Terapung Kepulauan Tihi-tihi dan Selangan, Bontang, Kalimantan Timur, Indonesia”, Cermin Dunia Kedokt., vol. 51, no. 2, pp. 67–70, https://doi.org/10.55175/cdk.v51i2.872, 2024.

[4] A. Rauf, “Perancangan Kampanye Sosial Pencegahan Penyakit Kulit Panu Melalui Media Poster”, Skripsi, Design Komunikasi Visual, Universitas Komputer Indonesia, 2019.

[5] M. Siannoto, “Diagnosis dan Tatalaksana Urtikaria”, Cermin Dunia Kedokt., vol. 44, no. 3, pp. 190–194, 2017.

[6] F. Nuraeni, Y. H. Agustin, dan E. N. Yusup, “Aplikasi Pakar Untuk Diagnosa Penyakit Kulit Menggunakan Metode Forward Chaining Di Al Arif Skin Care Kabupaten Ciamis”, Semnastekno Media Online, vol. 4, no. 1, pp. 3–4, 2016.

[7] N. S. Alk. ALEnezi, “A method of skin disease detection using image processing and machine learning”, Procedia Comput. Sci., vol. 163, pp. 85–92, 2019.

[8] M. I. Rizki, "Sistem Pakar Diagnosa Penyakit Kulit Menggunakan Na{\"\i}ve Bayes Berbasis Web", J. Transit, vol. 8, no. 4, pp. 27–34, 2020.

[9] S. Bhadula, S. Sharma, P. Juyal, and C. Kulshrestha, "Machine learning algorithms based skin disease detection", Int. J. Innov. Technol. Explor. Eng., vol. 9, no. 2, pp. 4044–4049, 2019.

[10] P. R. Kshirsagar, H. Manoharan, S. Shitharth, A. M. Alshareef, N. Albishry, and P. K. Balachandran, "Deep learning approaches for prognosis of automated skin disease", Life, vol. 12, no. 3, p. 426, 2022.

[11] M. M. Husein dan S. T. Dedi Gunawan, "Sistem Klasifikasi Penyakit Kulit pada Manusia Menggunakan Machine Learning Berbasis Android", Skripsi, Fakultas Ilmu Komunikasi dan Informatika > Teknik Informatika, Universitas Muhammadiyah Surakarta, 2024.

[12] R. Indriyanto and D. G. Purnama, "CI/CD Implementation Application Deployment Process Academic Information System (Case Study Of Paramadina University)", J. Indones. Sos. Teknol., vol. 4, no. 9, pp. 1503–1516, 2023.

[13] N. Nurhayati, "Implementation of Continuous Integration and Continuous Deployment (CI/CD) to Speed up the Automation Process of Software Delivery In the Production Process Using Node. Js, Docker, and React. Js", J. Info Sains Inform. dan Sains, vol. 14, no. 02, pp. 15–28, 2024.

[14] D. Kreuzberger, N. Kühl, and S. Hirschl, "Machine learning operations (mlops): Overview, definition, and architecture", IEEE access, vol. 11, pp. 31866–31879, 2023.

[15] M. Tan, "Efficientnet: Rethinking model scaling for convolutional neural networks", arXiv Prepr. arXiv1905.11946, 2019.