DECODE: JURNAL PENDIDIKAN TEKNOLOGI INFORMASI
P-ISSN: 2775-2984 | E-ISSN: 2775-1813

Vol. 5, No 3, November 2025, Hal: 826-838
Doi: http://dx.doi.org/10.51454/decode.v5i3.1196
https://journal.umkendari.ac.id/index.php/decode

Pengembangan Aplikasi Backend Web Service Berbasis RESTful API Untuk Aplikasi DermaScan Menggunakan Node.js

Ardi Wijaya[1], Andi Yahya[1*], RG Guntur Alam[1], Muhammad Imanullah[1], Gunawan[1]

[1]Program Studi Teknik Informatika, Universitas Muhammadiyah Bengkulu, Indonesia.

Artikel Info

Kata Kunci:

API;
Backend;
Perancangan Sistem;
RESTful API;
Web Service;

Keywords:

API;
Backend;
RESTful API;
System Design;
Web Service;

Riwayat Artikel:

Submitted: 21 Juni 2025
Accepted: 25 September 2025
Published: 26 September 2025

Abstrak: Aplikasi DermaScan merupakan aplikasi mobile untuk deteksi kanker kulit yang dibangun dengan arsitektur terpisah antara frontend dan backend. Untuk mendukung komunikasi antar sistem, dibutuhkan API berbasis REST yang memungkinkan pertukaran data secara efisien. Pada penelitian ini, backend dikembangkan menggunakan Node.js dengan metodologi waterfall yang meliputi tahap analisis kebutuhan, perancangan, implementasi, pengujian, dan pemeliharaan. Sistem diuji menggunakan metode black-box testing untuk mengukur fungsionalitas, serta load testing untuk menguji performa terhadap beban pengguna. Hasil black-box testing menunjukkan seluruh fitur berjalan sesuai dengan spesifikasi sistem, sedangkan load testing memperlihatkan API mampu menangani hingga 500 pengguna secara bersamaan dengan performa yang stabil. Backend ini dibangun menggunakan Express.js, terintegrasi dengan Google Cloud App Engine, Firestore sebagai basis data, serta Cloud Storage untuk penyimpanan gambar. Hasil pengembangan berupa API RESTful yang mendukung autentikasi, manajemen pengguna, artikel kesehatan, histori baca, dan diagnosis, yang siap diintegrasikan dengan aplikasi DermaScan. Penelitian ini diharapkan dapat menjadi acuan bagi pengembangan layanan kesehatan berbasis REST API.

***Abstrak*:** *DermaScan is a mobile application for skin cancer detection, developed with a decoupled architecture separating the frontend and backend. To enable efficient communication between systems, a REST-based API is required for data exchange. In this study, the backend was developed using Node.js with the waterfall methodology, which includes requirement analysis, system design, implementation, testing, and maintenance. The system was tested using black-box testing to evaluate functionality and load testing to assess performance under concurrent user demands. The black-box testing results confirmed that all features operated according to the system specifications, while load testing showed that the API could handle up to 500 concurrent users with stable performance. The backend was built using Express.js, integrated with Google Cloud App Engine, Firestore as the database, and Cloud Storage for image management. The outcome of this development is a RESTful API that supports authentication, user management, health articles, reading history, and diagnosis history, and is ready to be integrated with the DermaScan application. This study is expected to serve as a reference for the development of healthcare services based on REST API.*

Corresponding Author:
Andi Yahya
Email: andiyahya671@gmail.com

PENDAHULUAN

Pemanfaatan teknologi dibidang kesehatan sangatlah pesat dan telah memberikan dampak besar dalam memberikan sistem kesehatan yang baik (Fauzi et al., 2024)(Yuniartha et al., 2024). Implementasi aplikasi kesehatan berbasis *mobile* saat ini juga menjadi sangat populer karena perkembangan teknologi *mobile* yang semakin pesat (Salmani et al., 2020). DermaScan merupakan contoh dari aplikasi kesehatan yang berfungsi untuk mendeteksi kanker kulit. Kanker kulit merupakan salah satu jenis penyakit kanker yang mudah di deteksi sedari awal. Hal ini dikarenakan tanda-tanda penyakit tersebut dapat dengan mudah diidentifikasi pada area kulit. Aplikasi DermaScan dibuat untuk mendeteksi penyakit kanker kulit, dengan aplikasi ini pengguna cukup melakukan unggah gambar dan melakukan scan pada kulit yang memiliki tanda-tanda penyakit kanker kulit.

Untuk menjalankan fungsinya, DermaScan membutuhkan pertukaran data dengan server untuk mendukung beberapa fungsi pada aplikasi seperti manajemen pengguna, artikel kesehatan, dan histori diagnosis. Untuk mendukung fungsi-fungsi tersebut, DermaScan menerapkan arsitektur terpisah (*decoupled-architecture*). Sisi client atau frontend yang bertanggung jawab atas antarmuka dan interaksi pengguna, sementara aplikasi backend yang menangani logika bisnis, pemrosesan data dan komunikasi dengan database. Untuk menghubungkan kedua aplikasi tersebut maka diperlukanlah API (Application Programming Interface).

API atau Application Programming Interface adalah sekumpulan instruksi yang memfasilitasi komunikasi antara program client (aplikasi *mobile*) ke web service (aplikasi backend) (Guntara & Azkarin, 2023). Untuk menyediakan API tersebut maka diperlukanlah Aplikasi Backend Web Service, dimana API yang dikembangkan menggunakan arsitektur REST (Arianto & Susetyo, 2022). Tujuan dari API adalah memudahkan dalam melakukan koneksi antar sistem yang berbeda dalam konteks ini yaitu aplikasi frontend dan aplikasi backend (Ramadhan & Mulyati, 2024). Pemisahan kedua sistem ini juga akan memudahkan pemeliharaan jika dilakukan perbaikan dan pengembangan pada aplikasi (Tan et al., 2024).

REST atau Representational State Transfer dikemukakan oleh Roy Fielding dalam tesisnya pada tahun 2000. Arsitektur ini yang sering digunakan untuk berkomunikasi antara client-server melalui protokol HTTP. Namun, REST bukanlah sebuah protokol, format file atau framework, REST sebenarnya adalah sekumpulan aturan atau batasan yang disebut "Fielding Constraints" (Richardson & Amundsen, 2013). Roy Fielding mengemukakan batasan REST yang bersifat stateless, terdapat pemisahan client-server, *cacheable*, antarmuka yang konsisten dan seragam serta modular (Lauret, 2024). Hal ini yang menjadikannya solusi yang tepat untuk mengembangkan API berbasis web. Web services API yang sesuai dengan gaya arsitektur REST inilah yang kemudian disebut sebagai REST API atau RESTful API (Senduk et al., 2023). REST sebagai arsitektur API memiliki keunggulan dalam kecepatan, skalabilitas dan implementasinya yang mudah (Filiana et al., 2022). Beberapa peneliti sebelumnya telah mengembangkan berbagai sistem berbasis REST. Salah satunya adalah penelitian oleh (Fallo & Wibowo, 2023) yang membahas penerapan REST API dalam aplikasi reservasi dokter berbasis *mobile* android memperoleh hasil yang baik dimana semua fitur dapat berfungsi sesuai yang direncanakan. Selain itu, penelitian oleh (Falah et al., 2023) yang melakukan perancangan microservice berbasis REST menggunakan Node.js dan layanan Google Cloud Platform menghasilkan uji performa yang baik dengan rata-rata waktu tanggap sebesar 60,41 ms dan server mampu menerima rata-rata hits 40,07 per detik. Hasil dari beberapa penelitian ini menunjukkan performa yang sangat baik dari aplikasi yang menggunakan pendekatan REST.

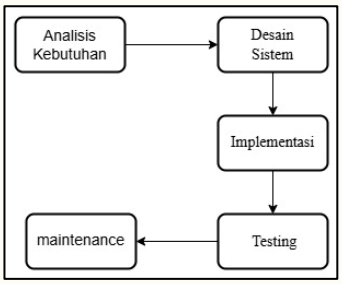
Untuk mengembangkan aplikasi backend berbasis REST maka diperlukan suatu bahasa pemrograman (Siahaan & Wijaya, 2024). Pada pengembangan ini akan menggunakan runtime pemrograman JavaScript yaitu Node.js. JavaScript awalnya hanya digunakan pada sisi browser atau client namun seiring dengan perkembangannya hadirlah runtime Node.js yang membuat beberapa kode JavaScript dapat di eksekusi di luar browser (Farchani et al., 2025). Node.js diketahui memiliki performa yang baik, beberapa penelitian telah membuktikan keunggulan REST API yang dikembangkan menggunakan Node.js. Penelitian berjudul "Perbandingan Rest API Menggunakan Node.js dan PHP Pada Aplikasi Pemilihan Umum" oleh (Haryadi et al., 2023) memperoleh hasil

perbandingan yaitu implementasi API yang dibuat menggunakan Node.js memiliki rata-rata hasil yang stabil dan konsisten dibanding dengan API yang dikembangkan menggunakan PHP. Sementara itu, penelitian oleh (Prayogi et al., 2020) menghasilkan kesimpulan bahwa implementasi REST API menggunakan Node.js memiliki performa yang lebih unggul saat menangani *request* sebanyak 1000 pengguna secara bersamaan dibanding aplikasi REST API yang dikembangkan menggunakan bahasa pemrograman PHP.

Oleh karena itu, penelitian ini bermaksud mengembangkan aplikasi backend berbasis REST menggunakan Node.js untuk menyediakan API yang dapat mendukung fungsionalitas aplikasi DermaScan de sehingga akan berkontribusi pada kualitas dan keandalan sistem secara keseluruhan. Selanjutnya hasil implementasi akan diuji menggunakan dua metode pengujian yaitu black-box testing untuk menguji fungsionalitas aplikasi dan load testing untuk menguji performa aplikasi. Sehingga penelitian ini diharapkan dapat berkontribusi dalam memberikan wawasan terkait pengembangan sistem backend web service berbasis REST bagi berbagai pihak pengembang perangkat lunak dan pihak terkait lainnya.

METODE

Penelitian ini menggunakan pendekatan dalam SDLC (Software Development Life Cycle). SDLC merupakan proses pengembangan atau mengubah suatu sistem perangkat lunak dengan menggunakan model-model dan metodologi untuk mengembangkan sistem perangkat lunak (Syahputri et al., 2022) (Elda et al., 2022). Salah satu metode SDLC yang umum digunakan dalam pengembangan software adalah Metode Waterfall. Metode waterfall atau metode air terjun adalah metode pengembangan perangkat lunak secara linear atau berurutan. Metode ini memiliki keunggulan yaitu praktis dan membuat software yang dikembangkan lebih kokoh (Hasanah & Untari, 2020). Proses pengerjaan-nya dilakukan secara bergantian inilah kemudian membentuk urutan pengerjaan yang menyerupai air terjun. Di dalam metode waterfall terdapat beberapa fase pengerjaan atau tahapan yaitu, analisis kebutuhan, desain sistem, implementasi, testing atau pengujian dan maintenance atau pemeliharaan aplikasi (Yuniartha et al., 2024).



Gambar 1. Metode Waterfall

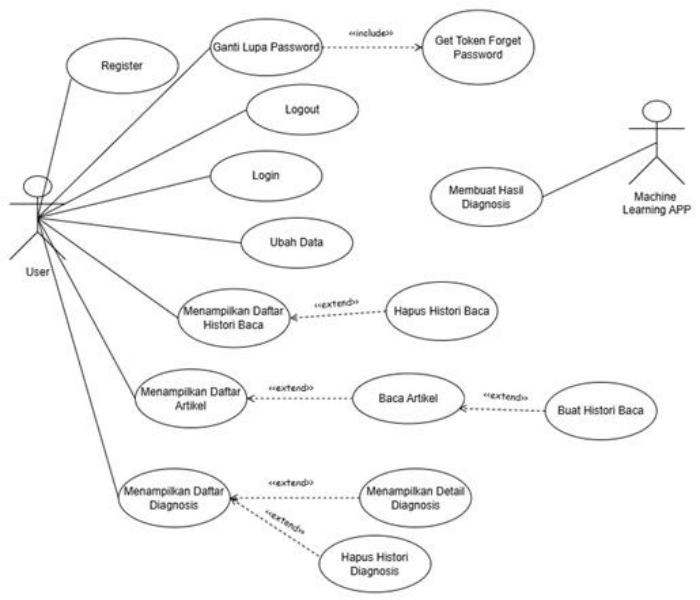
1. Analisis Kebutuhan

Pada tahap ini akan dilakukan analisis masalah berupa kebutuhan sistem atau fitur-fitur aplikasi DermaScan yang akan dibuat. Dilakukan analisis kompetitor untuk menentukan fitur dan sebagai bentuk peningkatan atau kelebihan dari aplikasi kompetitor. Analisis yang dilakukan terhadap beberapa aplikasi kompetitor seperti SkinVision, Miiskin, MoleMapper dan MoleScope telah ditemukan nilai unik dari aplikasi dermascan yaitu deteksi awal penyakit, dapat diakses semua orang dan dapat memberikan wawasan kepada pengguna.

Gambar 2. Perbandingan Aplikasi Kompetitor

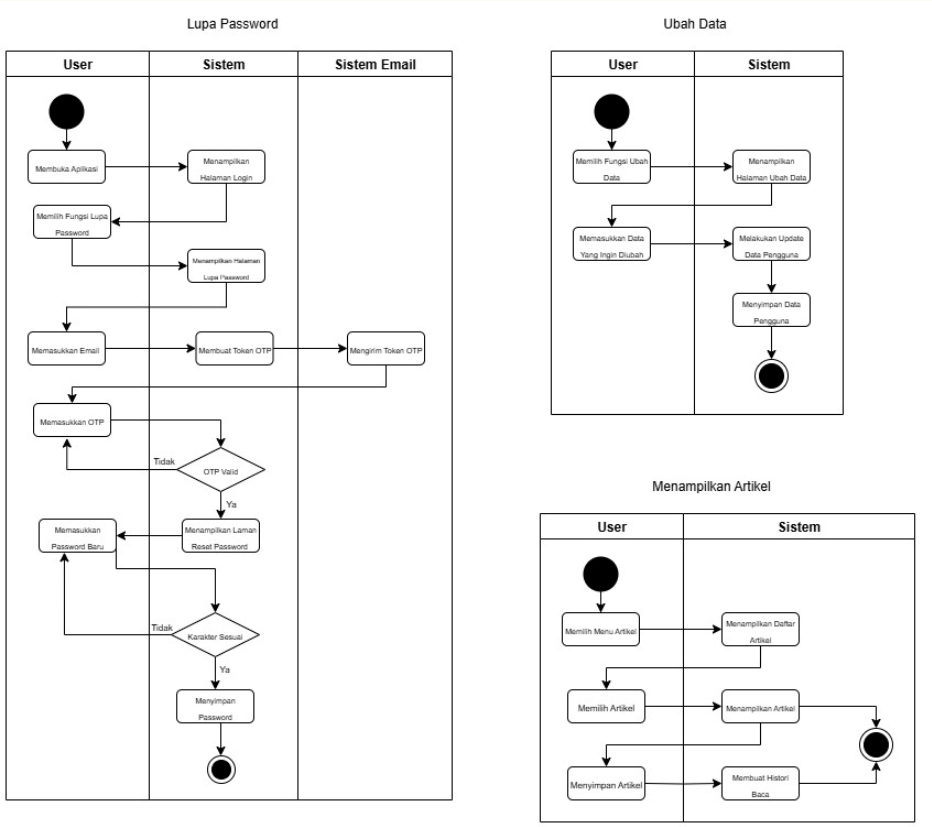
2. Desain Sistem

Setelah mendapatkan kebutuhan sistem berupa fitur yang akan dibuat. Selanjutnya, pada tahap ini akan dilakukan pemodelan sistem menggunakan diagram pada UML.



Gambar 3. Use Case Diagram

Use case diagram terdiri dari dua buah aktor yaitu user dan sistem lain yaitu sistem machine learning, user merupakan pengguna aplikasi DermaScan yang dapat melakukan beberapa aktivitas seperti login dan mendaftar akun (autentikasi dan manajemen pengguna), menampilkan daftar artikel, histori baca pengguna dan histori diagnosis. Sedangkan sistem backend lain yaitu machine learning dapat membuat dan menyimpan hasil diagnosis yang telah dilakukan.



Gambar 4. Activity Diagram

Setiap use case diagram akan dibuat ke dalam sebuah activity diagram. Activity Diagram akan digunakan untuk menggambarkan alur atau langkah-langkah dalam suatu proses, termasuk urutan aktivitas, aktivitas yang berjalan bersamaan, dan keputusan yang diambil. Aktivitas pada activity diagram pada sistem dimulai dengan simbol lingkaran dengan warna hitam pada Gambar 4, activity diagram menggambarkan alur aktivitas beberapa fitur yaitu:

Tabel 1. Penjelasan Activity Diagram

Aktivitas	Deskripsi
Register	Aktivitas registrasi dimulai ketika pengguna memilih menu pendaftaran dan mengisi data yang diperlukan. Setelah data dimasukkan, sistem akan melakukan validasi terhadap informasi yang diberikan, termasuk alamat email. Jika email yang dimasukkan sudah terdaftar, pengguna harus menggunakan email lain yang belum digunakan.
Login	Aktivitas login dimulai ketika pengguna membuka aplikasi dan memasukkan data login berupa email dan password. Jika data yang dimasukkan terdaftar dan valid, sistem akan memberikan token sesi, memungkinkan pengguna untuk berhasil masuk.
Logout	Pengguna dapat logout dengan memilih fitur logout dan mengonfirmasinya. Setelah itu, token sesi akan dihapus, dan pengguna harus login kembali untuk mengakses aplikasi.
Lupa Password	Pada halaman login, pengguna dapat memilih opsi lupa password dan diminta memasukkan email yang terdaftar di sistem. Jika email valid, sistem akan mengirimkan OTP ke email tersebut. Pengguna kemudian memasukkan OTP, dan jika benar, mereka dapat mengubah password, yang kemudian akan disimpan oleh sistem.
Ubah Data	Pengguna memilih fungsi ubah data, lalu memasukkan data baru yang valid pada halaman yang disediakan. Setelah itu, sistem akan menyimpan dan memperbarui data pengguna.
Menampilkan Histori Baca	Pengguna dapat meminta daftar histori baca, dan aplikasi akan menampilkan data histori tersebut. Selain itu, pengguna juga memiliki opsi untuk menghapus histori baca.
Menampilkan Artikel	Pengguna dapat meminta daftar artikel, dan aplikasi akan menampilkan daftar tersebut. Pengguna kemudian dapat memilih artikel untuk melihat detailnya serta menyimpan artikel ke dalam histori baca.
Menampilkan Hasil Diagnosis	Pengguna dapat meminta daftar hasil diagnosis, dan aplikasi akan menampilkan daftar tersebut. Pengguna dapat melihat detail diagnosis serta memiliki opsi untuk menghapus histori diagnosis.
Menyimpan Hasil Diagnosis	Sistem *backend* aplikasi *Machine Learning* dapat menyimpan hasil diagnosis pengguna.

3. Implementasi

Pada tahap ini, rancangan sistem yang telah dibuat akan diimplementasikan ke dalam baris kode program. Proses implementasi akan memanfaatkan runtime Node.js sebagai lingkungan eksekusi, serta Express.js sebagai library untuk pengembangan aplikasi. Aplikasi yang telah jadi akan di hosting pada layanan Google Cloud Provider (GCP), layanan yang digunakan adalah APP Engine.

4. Testing atau Pengujian

Setelah aplikasi dibuat maka aplikasi perlu diuji. Pengujian fungsionalitas sistem menggunakan metode black-box testing. Black-box testing adalah pengujian suatu sistem dengan tujuan untuk mengamati hasil input dan output dari sistem tersebut tanpa mengetahui struktur kode dari sistem itu sendiri (Nurdiana et al., 2024). Sedangkan untuk melakukan pengujian performa, dilakukan load testing dengan menggunakan Apache JMeter. Metode pengujian load testing memiliki kesamaan dengan stress testing dan performance testing, karena ketiganya termasuk dalam kategori non-functional testing (Jorgensen & DeVries, 2022). Dalam pengujian performa ini, setiap hasil tidak diperiksa secara mendetail, tetapi difokuskan untuk mengukur karakteristik sistem berdasarkan response time, throughput, dan error rate pada sistem yang diuji.

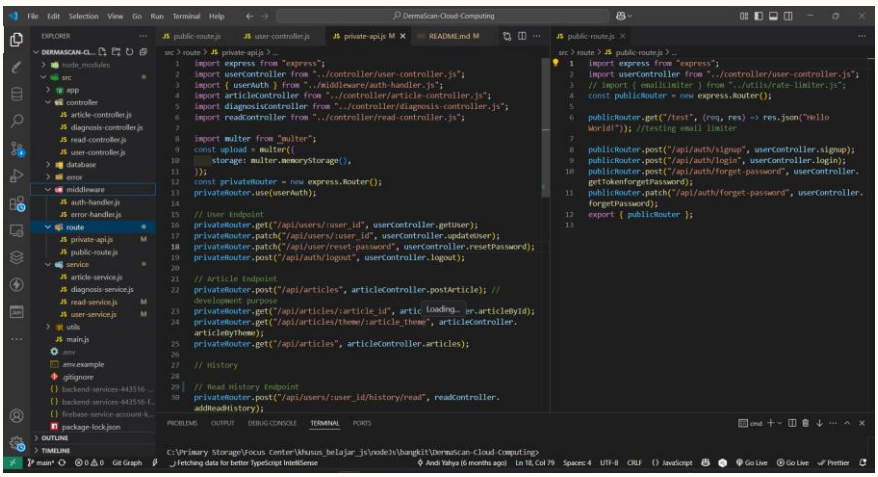
5. Maintenance atau Pemeliharaan Aplikasi

Pemeliharaan aplikasi bertujuan agar aplikasi yang dikembangkan sesuai dengan kebutuhan dan untuk meminimalisir bug.

HASIL DAN PEMBAHASAN

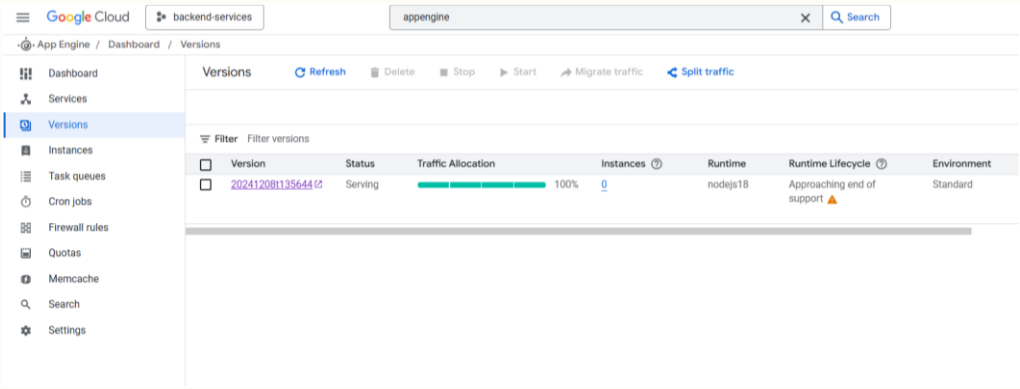
Pada tahap awal dilakukan analisis kebutuhan sistem dari aplikasi DermaScan. Dilakukan analisis kompetitor untuk menentukan fitur dan sebagai bentuk peningkatan atau kelebihan dari aplikasi kompetitor yang tersedia. Setelah mendapatkan kebutuhan sistem berupa fitur yang akan dibuat maka setiap fitur akan dibuat ke dalam diagram UML. Aplikasi backend ini dibangun di atas arsitektur cloud yang menggunakan Google App Engine dengan kelas lingkungan standar (Standard Environment). Instance yang digunakan adalah kelas F1 (default), yang menyediakan 384 MB memori dan 600 MHz CPU. Pemilihan kelas ini memberikan keseimbangan antara performa dan efisiensi biaya. Server ditempatkan di region asia-southeast2 (Jakarta) sehingga memastikan aplikasi nantinya akan memiliki latensi yang rendah. Untuk kebutuhan basis data, aplikasi ini menggunakan cloud firestore dalam konfigurasi *default*. cloud firestore dipilih karena kemampuannya dalam menangani data secara *real-time* dan skalabilitas horizontal yang sesuai untuk aplikasi modern berbasis cloud. Sementara itu, untuk penyimpanan data gambar, digunakan layanan Google Cloud Storage dengan kelas penyimpanan standar, yang juga ditempatkan di region asia-southeast2 (Jakarta).

Aplikasi ini dikembangkan menggunakan framework Express.js yaitu salah satu framework yang tersedia di Node.js untuk membangun server secara efisien. Proses pengembangan dilakukan menggunakan Visual Studio Code (VScode) sehingga akan mempercepat pengembangan aplikasi.



Gambar 5. Tampilan kode aplikasi pada kode editor VScode

Penelitian ini menghasilkan API berbasis arsitektur REST yang dibuat menggunakan Node.js dan telah berhasil di-*hosting* pada layanan Google Cloud Provider (GCP). Berikut ini adalah tampilan aplikasi backend berbasis Node.js pada *dashboard* APP Engine:

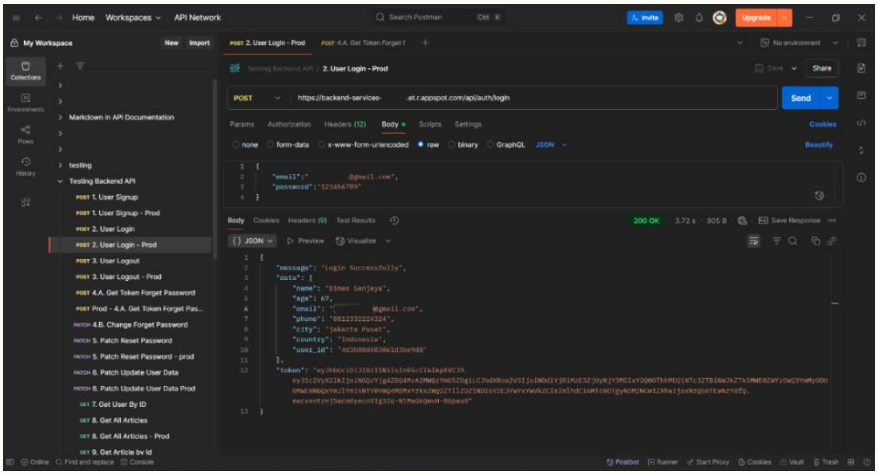


Gambar 6. Dashboard APP Engine

Tahap pengujian menggunakan dua buah metode pengujian yaitu black-box testing untuk menguji fungsionalitas aplikasi tanpa melihat struktur kode aplikasi, dan load testing yang berguna untuk menguji sistem dalam menangani beban pengguna atau permintaan dalam jumlah tertentu.

1. Black-Box Testing

Pengujian dilakukan menggunakan software postman, software ini digunakan untuk menguji tiap endpoint API yang telah dibuat. Pengujian aplikasi dilakukan di lingkungan uji coba (testing) dan production sehingga memastikan aplikasi benar-benar memenuhi spesifikasi aplikasi yang telah direncanakan dan menjadikan aplikasi minim bug atau kesalahan.



Gambar 7. Pengujian Black-box menggunakan Postman

Hasil pengujian black-box dapat dilihat pada tabel 2 berikut:

Tabel 2 Daftar Hasil Pengujian *Endpoint* API

No.	API	Kondisi	Hasil	Status
1	Signup	Memasukkan data dengan Benar	Registrasi berhasil	Sukses
		Memasukkan data yang tidak valid dan lengkap	Pesan kesalahan: memberi tahu data yang belum terisi dengan benar	Sukses
		Memasukkan data yang telah terdaftar	Pesan kesalahan: email telah digunakan	Sukses
2	Login	Memasukkan data yang valid	Login berhasil	Sukses

		Memasukkan data yang tidak valid	Pesan kesalahan: email dan password salah	Sukses
3	Logout	Melakukan logout dari aplikasi	Respon sukses: Logout berhasil	Sukses
		Pengguna *unauthorized* logout dari aplikasi	Pesan kesalahan: *Unauthorized*	Sukses
4	Mendapatkan *Token Forget Password*	Memasukkan email untuk permintaan token	Respon sukses: email berhasil dikirim	Sukses
		Memasukkan email yang tidak valid untuk permintaan token	Pesan kesalahan: *email required*	Sukses
5	Ubah Password	Memasukkan token valid dan password baru	Respon sukses: password berhasil diubah	Sukses
		Memasukkan token yang tidak valid dan password baru	Pesan kesalahan: *User not found*	Sukses
		Tidak Memasukkan token dan password baru	Pesan kesalahan: *Bad Request*	Sukses
6	Reset password	Ubah password dengan sesi valid	Respon sukses: password berhasil diubah	Sukses
		Ubah password dengan sesi tidak valid	Pesan kesalahan: *Unauthorized*	Sukses
		Ubah password dengan id pengguna yang tidak valid	Pesan kesalahan: *Unauthorized*	Sukses
7	Update Data	Memasukkan data baru dan sesi valid	Respon sukses: data berhasil di-*update*	Sukses
		Memasukkan data baru dan sesi tidak valid	Pesan kesalahan: *Unauthorized*	Sukses
		Memasukkan data baru dan id salah	Pesan kesalahan: *Unauthorized*	Sukses
		Memasukkan data tidak valid	Pesan kesalahan: *must be a valid data*	Sukses
8	Detail Pengguna	Akses data dengan sesi pengguna aktif	Respon sukses: data ditampilkan	Sukses
		Id pengguna tidak valid	Pesan kesalahan: *User not found*	Sukses
		Mencoba mengakses pengguna lain menggunakan *token* sesi yang berbeda	Pesan kesalahan: *Unauthorized*	Sukses
9	Daftar Artikel	Meminta daftar artikel dengan sesi valid	Respon sukses: data ditampilkan	Sukses
		Meminta daftar artikel dengan sesi yang tidak valid	Pesan kesalahan: *Unauthorized*	Sukses

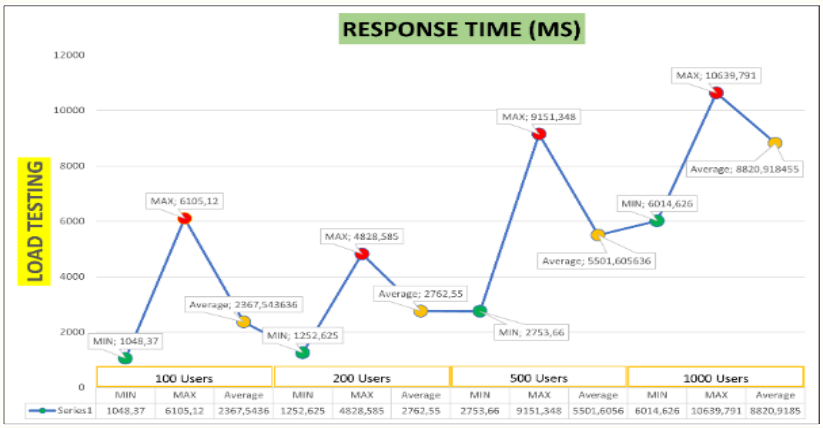
10	Detail Artikel	Mengakses sebuah artikel dengan sesi valid	Respon sukses: data ditampilkan	Sukses
		Mengakses sebuah artikel dengan sesi yang tidak valid	Pesan kesalahan: *Unauthorized*	Sukses
		Mengakses sebuah artikel yang tidak ada dengan sesi valid	Pesan kesalahan: *Article not found*	Sukses
11	Daftar Artikel Berdasarkan Tema	Meminta daftar artikel berdasar teman dengan sesi yang valid	Respon sukses: data ditampilkan	Sukses
		Meminta daftar artikel berdasar teman dengan sesi yang tidak valid	Pesan kesalahan: *Unauthorized*	Sukses
12	Menyimpan Histori Baca	Menyimpan artikel dengan sesi yang valid	Respon sukses: histori berhasil ditambahkan	Sukses
		Menyimpan artikel dengan sesi yang tidak valid	Pesan kesalahan: *Bad Request*	Sukses
		Menyimpan artikel yang tidak terdaftar	Pesan kesalahan: *Article not found*	Sukses
		Menyimpan artikel yang ditambahkan ke histori	Pesan kesalahan: *This article has already added to your history*	Sukses
13	Daftar Histori Baca	Melihat daftar histori baca pengguna dengan sesi valid	Respon sukses: data ditampilkan	Sukses
		Melihat daftar histori pengguna dengan sesi yang tidak valid	Pesan kesalahan: *Unauthorized*	Sukses
14	Hapus Histori Baca	Menghapus sebuah histori baca dengan sesi yang valid	Respon sukses: histori berhasil dihapus	Sukses
		Menghapus sebuah histori baca dengan sesi yang tidak valid	Pesan kesalahan: *Unauthorized*	Sukses
		Tidak memasukkan histori id	Pesan kesalahan: *Bad Request, specify article to delete*	Sukses
		Menghapus sebuah histori baca yang tidak ada	Pesan kesalahan: *Read History not found*	Sukses
15	Tambah Histori Diagnosis	Memasukkan image dan hasil test dengan sesi valid	Respon sukses: histori berhasil ditambahkan	Sukses
		Memasukkan image dan hasil test dengan sesi yang tidak valid	Pesan kesalahan: *Unauthorized*	Sukses
16	Daftar Histori Diagnosis	Melihat daftar histori diagnosis pengguna dengan sesi valid	Respon sukses: data ditampilkan	Sukses

		Melihat daftar histori diagnosis pengguna dengan sesi yang tidak valid	Pesan kesalahan: *Unauthorized*	Sukses
17	Detail Histori Diagnosis	Melihat detail histori diagnosis dengan sesi yang valid	Respon sukses: data ditampilkan	Sukses
		Melihat detail histori diagnosis dengan sesi yang tidak valid	Pesan kesalahan: *Unauthorized*	Sukses
		Melihat detail histori diagnosis yang tidak ada	Pesan kesalahan: *Diagnosis History not found*	Sukses
18	Hapus Histori Diagnosis	Menghapus sebuah histori pengguna dengan sesi yang valid	Respon sukses: data histori berhasil dihapus	Sukses
		Menghapus sebuah histori pengguna dengan sesi yang tidak valid	Pesan kesalahan: *Unauthorized*	Sukses
		Mencoba menghapus sebuah histori yang tidak ada	Pesan kesalahan: *Diagnosis History not found*	Sukses

Dari pengujian terhadap 18 endpoint API, menunjukkan bahwa sistem berhasil menangani *request* yang diberikan sesuai dengan desain spesifikasi API yang telah ditentukan.

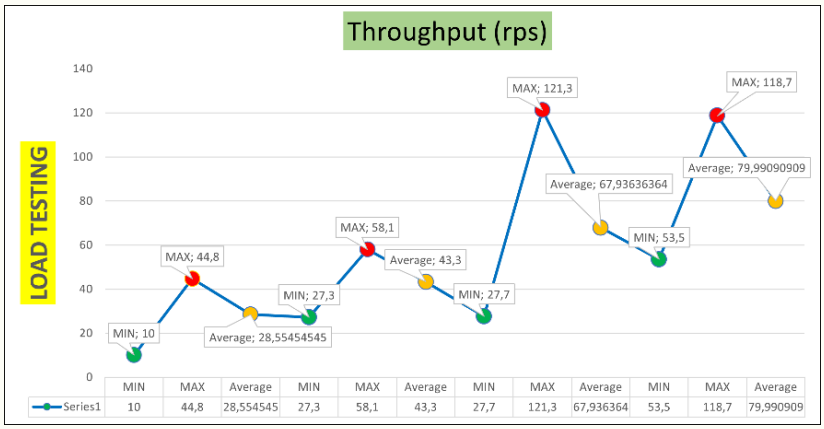
2. Load Testing

Pengujian dilakukan menggunakan Apache JMeter dalam mode non-GUI untuk memastikan pengujian berjalan secara optimal, sesuai dengan *best practice* dalam praktik load testing menggunakan JMeter. Skenario pengujian melibatkan pengiriman *request* dengan jumlah thread (*virtual users*) yang telah ditentukan, yaitu 100, 200, 500, dan 1000, yang dieksekusi dalam satu kali iterasi secara bersamaan. Pengujian ini berfokus pada beberapa indikator utama, yaitu rata-rata response time (waktu tanggap), rata-rata throughput (jumlah permintaan dalam jangka waktu tertentu), dan error rate (persentase permintaan gagal), guna mengevaluasi performa sistem di bawah beban yang berbeda. Berikut adalah grafik hasil pengujian load testing:



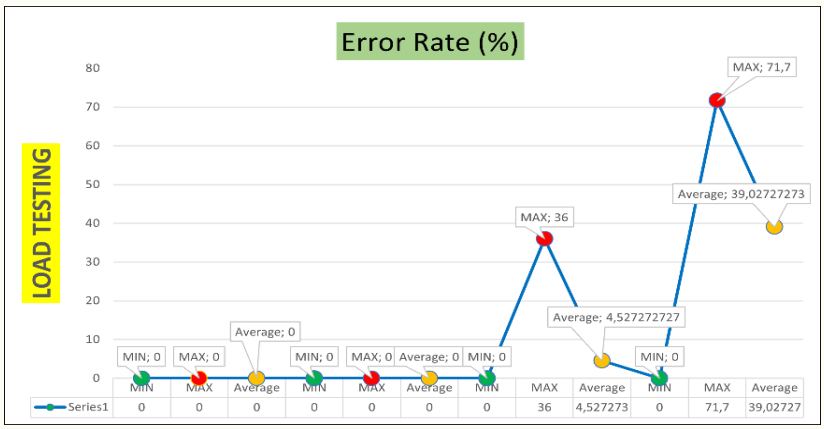
Gambar 8. Grafik Response Time

Berdasarkan hasil pengujian, performa API menunjukkan respons yang bervariasi tergantung pada jumlah pengguna yang mengakses secara bersamaan. Pada pengujian dengan 100 pengguna secara bersamaan, API memiliki rata-rata response time sebesar 2,3 detik, dengan response time terendah 1 detik dan tertinggi 6 detik. Throughput rata-rata yang dihasilkan adalah 28 RPS, dengan throughput maksimum mencapai 44,8 RPS.



Gambar 9. Grafik Throughput

Ketika jumlah pengguna meningkat menjadi 200 pengguna secara bersamaan, rata-rata response time sedikit meningkat menjadi 2,7 detik, dengan waktu respon terendah 1,2 detik dan maksimal 4,8 detik. Throughput rata-rata mencapai 43,3 RPS, dengan throughput maksimum sebesar 58 RPS. Pada tahap ini, sistem masih berjalan dengan normal tanpa adanya error (0% error rate). Namun, pada pengujian dengan 500 pengguna secara bersamaan, terjadi peningkatan signifikan dalam response time, dengan rata-rata 5,5 detik, response time terendah 2,7 detik, dan response time maksimal mencapai 9,1 detik. Throughput rata-rata yang diperoleh adalah 67 RPS, dengan throughput maksimum sebesar 121 RPS. Selain itu, mulai muncul error dengan rata-rata 4,5%, dan error rate tertinggi mencapai 36%.



Gambar 10. Grafik Error Rate

Ketika jumlah pengguna meningkat menjadi 1000 pengguna secara bersamaan, performa API semakin menurun dengan rata-rata response time mencapai 8,8 detik, response time terendah 6 detik, dan response time maksimal 10 detik. Throughput rata-rata yang dihasilkan adalah 80 RPS, dengan throughput maksimum sebesar 118 RPS. Error rate meningkat secara signifikan, dengan rata-rata 39% dan error tertinggi mencapai 71,7%, menunjukkan bahwa sistem tidak mampu menangani lonjakan beban yang tinggi secara optimal.

Secara keseluruhan, sistem berjalan dengan stabil tanpa error hingga 200 pengguna secara bersamaan, namun mulai mengalami degradasi performa dan munculnya error ketika jumlah pengguna mencapai 500, dengan tingkat persentase error yang semakin meningkat pada 1000 pengguna.

KESIMPULAN

Aplikasi backend berbasis REST menggunakan Node.js telah berhasil dibuat setelah melalui berbagai tahap pengembangan. API telah siap untuk diintegrasikan dengan aplikasi frontend dan sistem lain. Aplikasi backend ini telah menyediakan API yang dapat memfasilitasi komunikasi server dengan aplikasi frontend DermaScan dengan telah menyediakan 18 endpoint yang mencakup beberapa fungsi aplikasi yaitu autentikasi dan manajemen pengguna, daftar artikel, histori baca pengguna dan histori diagnosis. Tiap endpoint juga telah melalui tahapan pengujian yaitu black-box testing dan load testing. Hasil pengujian black-box menunjukkan aplikasi yang dirancang dan diimplementasikan menggunakan Node.js dapat berjalan sesuai spesifikasi API yang diharapkan. Selain itu, hasil pengujian menggunakan load testing menunjukkan performa yang cukup baik, hasil pengujian menunjukkan bahwa aplikasi dapat menangani hingga 500 pengguna secara bersamaan dengan performa yang cukup baik. Keterbatasan spesifikasi layanan cloud yang digunakan menjadi penyebab utama performa aplikasi kurang maksimal dalam melayani lonjakan request secara mendadak. Aplikasi ini akan terus dikembangkan seiring dengan perkembangan aplikasi DermaScan sehingga benar-benar dapat memenuhi kebutuhan pengguna. Namun, penelitian ini masih memiliki keterbatasan pada aspek pengujian skala besar dan performa layanan cloud. Oleh karena itu, untuk pengembangan selanjutnya, disarankan menggunakan layanan cloud dengan spesifikasi lebih tinggi serta melakukan pengujian lebih luas seperti pemantauan parameter kinerja server untuk memastikan skalabilitas dan keandalan sistem.

DAFTAR PUSTAKA

Arianto, O. D., & Susetyo, Y. A. (2022). Penerapan Restful Web Service Dengan Framework Laravel Untuk Pembangunan Sistem Informasi Manajemen Sumber Daya Manusia. *JIPI: Jurnal Ilmiah Penelitian dan Pembelajaran Informatika, 7(2).* https://doi.org/10.29100/jipi.v7i2.2870

Elda, E. S., Mulyono, H., & Pernanda, A. Y. (2022). Perancangan Sistem Informasi Layanan Pengaduan Badan Eksekutif Mahasiswa Berbasis Web. *Decode: Jurnal Pendidikan Teknologi Informasi*, *3*(1), 1–11. https://doi.org/10.51454/decode.v3i1.67

Falah, R. F., Komarudin, M., & Pratama, M. (2023). Perancangan Microservice Berbasis Rest Api Pada Google Cloud Platform Menggunakan Nodejs Dan Python. *JITET (Jurnal Informatika dan Teknik Elektro Terapan)*, *11*. http://dx.doi.org/10.23960/jitet.v11i3%20s1.3506

Fallo, A. C., & Wibowo, A. P. (2023). Penerapan REST API Untuk Aplikasi Reservasi Dokter Praktik Berbasis Android (Studi Kasus: Klinik dr. Candra Safitri). *Teknika*, *12*(2), 106–114. https://doi.org/10.34148/teknika.v12i2.615

Farchani, S. B., Hermanto, N., & Kusuma, B. A. (2025). Implementasi Rest Api Dalam Pengembangan Backend Inventory Peminjaman. *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)*, *10*(2), 1404–1413. https://doi.org/10.29100/jipi.v10i2.6249

Fauzi, M. R., Saimi, S., & Fathoni, F. (2024). Tantangan dan Solusi Administrasi Kesehatan di Era Digital (Tinjauan Literature Review atas Implementasi Teknologi). *AL-MIKRAJ Jurnal Studi Islam Dan Humaniora (E-ISSN 2745-4584)*, *5*(01), 1093–1103. https://doi.org/10.37680/almikraj.v5i01.6219

Filiana, A., Nila, M., Rini, A., Prabawati, A. G., & Samat, R. A. (2022). Pengembangan Rest Api Untuk Informasi Pasar Tradisional Di Kota Yogyakarta Dengan Metode Incremental. *SINTECH* https://doi.org/https://doi.org/10.31598/sintechjournal.v5i1.1060

Guntara, R. G., & Azkarin, V. (2023). Implementasi dan Pengujian REST API Sistem Reservasi Ruang Rapat dengan Metode Black Box Testing. *Jurnal Minfo Polgan*, *12*(1), 1229–1238. https://doi.org/10.33395/jmp.v12i1.12691

Haryadi, H. L., Sujjada, A., & Simatupang, D. S. (2023). Perbandingan Rest Api Menggunakan Node Js Dan Php Pada Aplikasi Pemilihan Umum. *Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)*, *8*(2), 460–468. https://dx.doi.org/10.30645/jurasik.v8i2.631

Hasanah, F. N., & Untari, R. S. (2020). Buku Ajar Rekayasa Perangkat Lunak. *UMSIDA PRESS*. https://doi.org/10.21070/2020/978-623-6833-89-6

Jorgensen, P., & DeVries, B. (2022). Software Testing; A Craftsman's Approach. http://dx.doi.org/10.1201/9781003168447

Lauret, A. (2024). *The Design of Web APIs, Second Edition*. Manning Publications.

Nurdiana, D., Aprijani, D. A., Amastini, F., Maulana, M. R., & Utama, Moh. R. P. A. (2024). Pengembangan Sistem Informasi Pengelolaan Pembimbing Lapangan Praktik Kerja Lapangan (PKL) Prodi Sistem Informasi Universitas Terbuka. *Decode: Jurnal Pendidikan Teknologi Informasi*, *4*(2), 418–436. https://doi.org/10.51454/decode.v4i2.433

Prayogi, A. A., Niswar, M., Indrabayu, & Rijal, M. (2020). Design and Implementation of REST API for Academic Information System. *IOP Conference Series: Materials Science and Engineering*, *875*(1). https://doi.org/10.1088/1757-899X/875/1/012047

Ramadhan, A., & Mulyati, S. (2024). Implementation Of Json Web Token In The Development Of Village Monograph Database Based On Restapi. *Jurnal Teknik Informatika (Jutif)*, *5*(6), 1849–1860. https://doi.org/10.52436/1.jutif.2024.5.6.4028

Richardson, L., & Amundsen, M. (2013). *RESTful Web APIs*. O'Reilly Media.

Salmani, H., Ahmadi, M., & Shahrokhi, N. (2020). The Impact of Mobile Health on Cancer Screening: A Systematic Review. *Cancer Informatics (Vol. 19).* https://doi.org/10.1177/1176935120954191

Senduk, F. X., Najoan, X. B. N., & Sompie, S. R. U. A. (2023). Pengembangan Arsitektur Microservices dengan RESTful API Gateway menggunakan Backend-for-frontend Pattern pada Portal Akademik Perguruan Tinggi. *Jurnal Teknik Informatika, 18(1).* https://doi.org/10.35793/jti.v18i1.50402

Siahaan, M., & Wijaya, R. W. (2024). Performance Comparison Between Laravel and ExpressJs Framework Using Apache JMeter. *Journal Of Informatics And Telecommunication Engineering*, *7*(2), 545–554. https://doi.org/10.31289/jite.v7i2.10571

Syahputri, W. D., Pratama, A., & Pernanda, A. Y. (2022). Perancangan Sistem Informasi Program Kerja Organisasi Kemahasiswaan Berbasis Web. *Decode: Jurnal Pendidikan Teknologi Informasi*, *3*(1), 22–29. https://doi.org/10.51454/decode.v3i1.68

Tan, R., Wijanto, M. C., & Lieshiana, C. (2024). PerancanganAplikasi Orientasi Mahasiswa Barudengan Android, Laravel Framework,dan Lean Touch. *Jurnal Teknik Informatika Dan Sistem Informasi (JuTISI*. http://dx.doi.org/10.28932/jutisi.v9i3.7945

Yuniartha, D., Sari, I., & Sufyana,. C. M. (2024). Perancangan Sistem Informasi Pelaporan 20 Besar Penyakit Pasien BPJS Rawat Jalan Menggunakan Metode Waterfall. *Decode: Jurnal Pendidikan Teknologi Informasi*, *4*(2), 609–620. https://doi.org/10.51454/decode.v4i2.588