TUGAS AKHIR

PENGEMBANGAN LIBRARY FLUTTER UNTUK OTOMATISASI INTEGRASI API BERBASIS SWAGGER MENGGUNAKAN METODE EXTREME PROGRAMMING

DISUSUN OLEH:

RIFKY TYO RAMADHANI SANTOSO

NIM. 22106050034

PROGRAM STUDI INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA

YOGYAKARTA

2026

LEMBAR PENGESAHAN

KEMENTERIAN AGAMA
UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Marsda Adisucipto Telp. (0274) 540971 Fax. (0274) 519739 Yogyakarta 55281

PENGESAHAN TUGAS AKHIR

Nomor : B-789/Un.02/DST/PP.00.9/05/2026

Tugas Akhir dengan judul : Pengembangan Library Flutter Untuk Otomatisasi Integrasi API Berbasis Swagger Menggunakan Metode Extreme Programming

yang dipersiapkan dan disusun oleh:

Nama	: RIFKY TYO RAMADHANI SANTOSO
Nomor Induk Mahasiswa	: 22106050034
Telah diujikan pada	: Rabu, 22 April 2026
Nilai ujian Tugas Akhir	: A

dinyatakan telah diterima oleh Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta

TIM UJIAN TUGAS AKHIR

Ketua Sidang

Muhammad Galih Wonoseto, M.T.
SIGNED

Valid ID: 69f972346b80b

Penguji I

Ir. Maria Ulfah Siregar, S.Kom., MIT., Ph.D.
SIGNED

Valid ID: 69f26030d6f8c

Penguji II

Eko Hadi Gunawan, M.Eng.
SIGNED

Valid ID: 69f96f10a26cc

Yogyakarta, 22 April 2026
UIN Sunan Kalijaga
Dekan Fakultas Sains dan Teknologi

Prof. Dr. Dra. Hj. Khurul Wardati, M.Si.
SIGNED

Valid ID: 69f9695dfab39

LEMBAR PERNYATAAN

LEMBAR PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini:

Nama	: Rifky Tyo Ramadhani Santoso
NIM	22106050034
Program Studi	: Informatika
Fakultas	: Sains dan Teknologi

Menyatakan dengan sesungguhnya, bahwa skripsi saya yang berjudul **"Pengembangan Libray Flutter Untuk Otomatisasi Integrasi API Berbasis Swagger Menggunakan Metode Extreme Programming"** merupakan hasil pekerjaan penulis sendiri sepanjang pengetahuan penulis, bukan duplikasi atau saduran dari karya orang lain kecuali bagian tertentu yang penulis ambil sebagai bahan acuan. Apabila terbukti pernyataan ini tidak benar, sepenuhnya menjadi tanggung jawab penulis.

Yogyakarta, 06 April 2026
Yang membuat pernyataan,

Rifky Tyo Ramadhani Santoso
NIM. 22106050034

LEMBAR PERSETUJUAN

Universitas Islam Negeri Sunan Kalijaga FM-UINSK-BM-05-03/R0

SURAT PERSETUJUAN SKRIPSI/TUGAS AKHIR

Hal : Persetujuan Skripsi / Tugas Akhir
Lamp : -

Kepada
Yth. Dekan Fakultas Sains dan Teknologi
UIN Sunan Kalijaga Yogyakarta
di Yogyakarta

Assalamu'alaikum wr. wb.

Setelah membaca, meneliti, memberikan petunjuk dan mengoreksi serta mengadakan perbaikan seperlunya, maka kami selaku pembimbing berpendapat bahwa skripsi Saudara:

Nama	:	Rifky Tyo Ramadhani Santoso
NIM	:	22106050034
Judul Skripsi	:	Pengembangan Library Flutter Untuk Otomatisasi Integrasi API Berbasis Swagger Menggunakan Metode Extreme Programming

sudah dapat diajukan kembali kepada Program Studi Informatika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata Satu dalam Program Studi Informatika.

Dengan ini kami mengharap agar skripsi/tugas akhir Saudara tersebut di atas dapat segera dimunaqasyahkan. Atas perhatiannya kami ucapkan terima kasih.

Wassalamu'alaikum wr. wb.

Yogyakarta, 06 April 2026
Pembimbing,

Muhammad Galih Wonoseto, M.T.
NIP. 19901113 201903 1 012

INTISARI

Pengembangan aplikasi *mobile* modern kerap menggunakan *Application Programming Interface* (API) sebagai sumber data. Namun, proses integrasi API pada Flutter masih dilakukan secara manual sehingga menimbulkan penulisan kode berulang, potensi kesalahan, dan kurangnya konsistensi struktur kode. Penelitian ini bertujuan untuk mengembangkan dan mengevaluasi library Flutter bernama Swagen yang mampu mengotomatisasi integrasi API berbasis Swagger/ OpenAPI melalui pendekatan *code generation*. Metode yang digunakan adalah *Extreme Programming*, sedangkan pengujian dilakukan menggunakan *Black Box Testing* serta terdapat evaluasi sistem menggunakan *Technology Acceptance Mode* (TAM). Hasil penelitian menunjukkan bahwa seluruh fitur pada Swagen berjalan sesuai dengan spesifikasi yang telah ditetapkan. Selain itu, hasil evaluasi TAM menunjukkan tingkat penerimaan yang baik hingga sangat baik dari pengguna, dengan nilai rata-rata *Perceived Ease of Use* sebesar 4,13 (sangat baik), *Perceived Usefulness* sebesar 4,35 (sangat baik), *Attitude Toward Using* sebesar 4,23 (sangat baik), *Behavioral Intention to Use* sebesar 4,28 (sangat baik), *serta Actual Use* sebesar 3,67 (baik). Dengan demikian, Swagen mampu meningkatkan efisiensi pengembangan aplikasi Flutter, mengurangi pekerjaan manual, serta menghasilkan struktur kode yang lebih konsisten sesuai dengan prinsip *Clean Architecture*.

Kata Kunci: Library, Flutter, Code Generation, Swagger/OpenAPI, *Extreme Programming*, TAM

ABSTRACT

Modern mobile application development frequently utilizes Application Programming Interfaces (APIs) as a primary data source. However, the process of API integration in Flutter *is still commonly performed manually, resulting in repetitive code writing, increased potential for errors, and a lack of consistency in code structure. This study aims to develop and evaluate a* Flutter *library called* Swagen*, which is capable of automating API integration based on Swagger/OpenAPI through a code generation approach. The development method used is Extreme Programming, while testing is conducted using Black Box Testing and system evaluation is carried out using the Technology Acceptance Model (TAM). The results show that all features of* Swagen *function according to the specified requirements. Furthermore, the TAM evaluation results indicate a good to very good level of user acceptance, with an average score of 4.13 (very good) for Perceived Ease of Use, 4.35 (very good) for Perceived Usefulness, 4.23 (very good) for Attitude Toward Using, 4.28 (very good) for Behavioral Intention to Use, and 3.67 (good) for Actual Use. Thus,* Swagen *is able to improve the efficiency of* Flutter *application development, reduce manual work, and produce a more consistent code structure in accordance with Clean Architecture principles.*

Keywords: Library, Flutter, Code Generation, Swagger/OpenAPI, *Extreme Programming*, TAM

MOTTO

“ Keluhan demi keluhan, Segera kugilas perlahan.”

Gemilang Perunggu

“Angan dan pertanyaan, waktu yang menjawabnya.”

Mata Air Hindia

“Tidak masalah seberapa lambat kau berjalan asalkan kau tidak berhenti.”

Confucius

HALAMAN PERSEMBAHAN

Dengan rasa syukur yang mendalam kepada Allah Yang Maha Pengasih lagi Maha Penyayang, saya mempersembahkan Tugas Akhir ini sebagai wujud penghormatan dan ungkapan kasih yang tulus. Semoga karya ini memberikan manfaat serta bernilai sebagai amal kebaikan. Tugas Akhir ini saya dedikasikan dengan penuh cinta dan rasa hormat kepada:

1. Ibu tercinta, Maryati, yang senantiasa memberikan kasih sayang, doa, dukungan, serta pengorbanan yang tiada henti dalam setiap langkah hidup saya.
2. Bapak tercinta, Imam Wahyu Santoso, yang selalu menjadi teladan, memberikan semangat, serta dukungan moral dan materiil demi keberhasilan saya.
3. Kakak dan adik saya, Destia Mareta Dyah Santoso dan Dhanda Aptyuril Santoso, yang selalu memberikan dukungan, kebersamaan, dan motivasi dalam setiap proses yang saya jalani.
4. Sahabat dan teman seperjuangan, yang telah menemani, membantu, serta memberikan semangat dan kebersamaan selama masa perkuliahan hingga penyusunan Tugas Akhir ini.

KATA PENGANTAR

Puji syukur kehadirat Allah SWT atas segala rahmat, hidayah, dan karunia-Nya, sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "Pengembangan Library Flutter untuk Otomatisasi Integrasi API Berbasis Swagger Menggunakan Metode Extreme Programming" dengan baik dan tepat waktu.

Tugas Akhir ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana pada Program Studi Informatika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Sunan Kalijaga Yogyakarta. Dalam proses penyusunan Tugas Akhir ini, penulis menyadari bahwa tidak terlepas dari bantuan, bimbingan, serta dukungan dari berbagai pihak.

Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Prof. Noorhaidi Hasan S.Ag., M.A., M.Phil., Ph.D. selaku Rektor UIN Sunan Kalijaga Yogyakarta.
2. Prof. Dr. Dra. Hj. Khurul Wardati, M.Si. selaku Dekan Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta.
3. Bapak Dr. Muhammad Mustakim, S.T. M.T. selaku Ketua Program Studi Informatika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta.
4. Dr. Agus Mulyanto, S.Si., M.Kom., ASEAN Eng. selaku dosen penasihat akademik yang telah memberikan arahan dan bimbingan selama masa perkuliahan penulis.
5. Muhammad Galih Wonoseto, M.T. selaku dosen pembimbing yang telah memberikan bimbingan, masukan yang berharga, serta dukungan penuh dengan penuh kesabaran dalam setiap tahapan penyusunan Tugas Akhir ini.
6. Seluruh dosen dan karyawan Program Studi Informatika UIN Sunan Kalijaga Yogyakarta yang telah memberikan ilmu, bantuan, serta pelayanan selama penulis menempuh pendidikan.
7. Kedua orang tua penulis, Imam Wahyu Santoso dan Maryati, yang senantiasa memberikan doa, kasih sayang, dukungan, serta pengorbanan yang tiada henti demi keberhasilan penulis.
8. Kakak dan adik penulis, Destia Mareta Dyah Santoso dan Dhanda Aptyuril Santoso, yang selalu memberikan semangat, dukungan, dan motivasi dalam setiap proses yang dilalui penulis.

9. Sahabat serta teman-teman seperjuangan yang telah memberikan dukungan, bantuan, serta kebersamaan selama proses perkuliahan hingga penyusunan Tugas Akhir ini.
10. Semua pihak yang tidak dapat disebutkan satu per satu yang telah memberikan bantuan, dukungan, dan doa dalam penyelesaian Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki berbagai keterbatasan, baik dari segi kualitas maupun kelengkapan materi yang disajikan. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang bersifat membangun guna menyempurnakan karya ini. Penulis juga berharap Tugas Akhir ini dapat memberikan manfaat serta menjadi dasar untuk pengembangan lebih lanjut di masa yang akan datang.

Yogyakarta, 06 April 2026

Penulis,

Rifky Tyo Ramadhani Santoso

NIM.22106050034

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pengembangan aplikasi *mobile* modern semakin banyak bergantung pada layanan berbasis *Application Programming Interface* (API) sebagai sumber data utama. API berperan penting dalam memungkinkan aplikasi *mobile* berkomunikasi dengan sistem *backend* secara terstruktur [1]. Dengan adanya API, proses pertukaran data dapat dilakukan secara terstandarisasi, sehingga mendukung pengembangan aplikasi yang bersifat modular, mudah dikembangkan, dan dapat diintegrasikan lebih mudah dengan alat pihak ketiga [2].

Flutter sebagai salah satu *framework* pengembangan aplikasi *mobile* lintas platform yang populer menyediakan kemudahan dalam membangun antarmuka pengguna yang responsif dan konsisten [3], [4]. Flutter memungkinkan pengembang untuk membangun aplikasi Android dan iOS dengan satu basis kode, sehingga meningkatkan efisiensi pengembangan [5]. Namun demikian, meskipun unggul dalam aspek *user interface,* proses integrasi API pada Flutter masih sering melakukan langkah-langkah manual yang memakan waktu dan tenaga.

Berdasarkan survei yang dilakukan peneliti, mayoritas responden menyatakan bahwa integrasi API di Flutter masih dilakukan secara manual dan memerlukan waktu yang cukup besar, terutama dalam proses parsing JSON dan pembuatan berbagai komponen seperti *data class*, *repository*, model *request/response*, serta *remote data source* untuk setiap endpoint. Secara umum, sekitar 85%–90% responden menjawab *Ya* pada pertanyaan terkait proses manual, kebutuhan pembaruan kode akibat perubahan API, serta kompleksitas integrasi. Proses yang berulang ini menimbulkan *boilerplate code* berlebih, meningkatkan risiko kesalahan, serta menyebabkan ketidakkonsistenan struktur kode, khususnya pada proyek berskala besar. Selain itu, sekitar 80%–85% responden juga menyatakan pernah mengalami error akibat perubahan API dan menilai bahwa tools seperti *Swagger/OpenAPI* dan *code generator* yang ada belum sepenuhnya efisien. Di sisi lain, lebih dari 90% responden menyatakan kebutuhan terhadap solusi otomatisasi serta ketertarikan terhadap tools seperti *Swagen*. Oleh karena itu, dibutuhkan solusi untuk mengotomatisasi integrasi API agar proses *development* menjadi lebih cepat, konsisten, dan minim kesalahan, sejalan dengan penelitian Mingxing Liu *et al*. yang menunjukkan bahwa pendekatan *code generation* dapat meningkatkan efisiensi dan kualitas pengembangan perangkat lunak [6].

Di sisi lain, Swagger atau OpenAPI Specification telah menjadi standar umum dalam pendokumentasian API [7]. Dokumen Swagger menyediakan definisi lengkap dan terstruktur mengenai *endpoint*, parameter, skema data, serta format *response* API yang memungkinkan interoperabilitas sistem secara konsisten [8] [9]. Informasi tersebut bersifat *machine-readable,* sehingga sangat memungkinkan untuk dimanfaatkan dalam proses otomatisasi, khususnya dalam pembuatan kode klien (*client code*) yang sesuai dengan spesifikasi API yang tersedia [10].

Namun, pada ekosistem Flutter saat ini, dukungan terhadap konversi otomatis dari dokumen Swagger menjadi API *client* masih relatif terbatas. Pengembang Flutter masih harus melakukan pemetaan data secara manual meskipun dokumentasi API telah tersedia secara lengkap. Hal ini menyebabkan potensi ketidaksesuaian antara implementasi aplikasi dengan dokumentasi API, terutama ketika API mengalami perubahan struktur atau penambahan fitur secara mendadak. Kondisi tersebut berdampak pada menurunnya efisiensi proses pengembangan aplikasi, khususnya dalam proyek berskala besar atau proyek yang memiliki frekuensi perubahan API yang tinggi. Proses manual yang berulang juga meningkatkan risiko terjadinya kesalahan dalam pemetaan model data dan *respons* API, yang dapat menyebabkan kesalahan saat *runtime* dan menurunkan kualitas aplikasi secara keseluruhan [6].

Untuk menyelesaikan permasalahan tersebut, peneliti melakukan pengembangan sebuah *library* Flutter bernama Swagen yang berfungsi untuk mengotomatisasi proses integrasi API dengan menghasilkan *data layer, domain layer*, dan *presentation layer* secara otomatis berdasarkan dokumen Swagger/OpenAPI. Pada pengembangan library ini, peneliti menggunakan metode *Extreme Programming* (XP) yang termasuk ke dalam *Agile Methods*. Berdasarkan artikel Fachrurozi *et al*., *Extreme Programming* memiliki empat tahap utama, yaitu *planning, design, coding*, dan *testing* [11]. Dalam konteks penelitian ini, tahapan tersebut diterjemahkan ke dalam proses analisis kebutuhan terhadap struktur dokumen Swagger, perancangan arsitektur generator, implementasi kode secara iteratif, serta pengujian terhadap setiap fitur generator yang dikembangkan, sehingga menjadi dasar perumusan masalah sekaligus menggambarkan tujuan yang ingin dicapai pada setiap fase pengembangan.

Pemilihan metode *Extreme Programming* didasarkan pada karakteristik pengembangan *library* yang sangat memungkinkan terjadinya perubahan kebutuhan maupun desain, seperti perubahan struktur *schema* API, jenis *generator* yang dibutuhkan, atau format keluaran yang diharapkan oleh *developer*. Selain itu, pengguna *library* (*developer* Flutter) cenderung membutuhkan penyesuaian fitur secara cepat dan berulang. Hal ini sejalan dengan pendapat Rina Juniar, Imam Ma'ruf Nugroho, dan Yusuf Muhyidin (2025) dalam penelitian berjudul "*E-*

Commerce Website Design Using Extreme Programming Method (Case Study: UMKM Bolu Ara)", yang menyatakan bahwa XP bersifat iteratif dan fleksibel, cocok untuk pengembangan perangkat lunak berskala kecil hingga menengah, serta mampu menyesuaikan kebutuhan pengguna yang dinamis [12].

Dengan dikembangkannya *library* Flutter untuk pembuatan otomatis struktur API dari dokumen Swagger menggunakan metode *Extreme Programming*, diharapkan solusi ini mampu meningkatkan efisiensi pengembangan aplikasi Flutter. Selain itu, library ini juga diharapkan dapat menjadi alternatif yang efektif bagi pengembang dalam mengintegrasikan API secara lebih cepat, konsisten, dan minim kesalahan, sehingga mendukung praktik pengembangan perangkat lunak yang lebih adaptif dan berkelanjutan.

1.2 Rumusan Masalah

1. Bagaimana merancang dan membangun *library* Flutter (Swagen) untuk mengotomatisasi integrasi API berbasis Swagger/OpenAPI melalui pendekatan code generation?
2. Bagaimana implementasi prinsip *Clean Architecture* (layer *data*, *domain*, dan *presentation*) serta penerapan *dependency injection* pada *library* yang dikembangkan?
3. Bagaimana menguji fungsionalitas *library* menggunakan metode *Black Box Testing*?
4. Bagaimana tingkat penerimaan pengguna terhadap library Swagen berdasarkan pendekatan *Technology Acceptance Model* (TAM)?

1.3 Tujuan

Tujuan dari Penelitian ini adalah sebagai berikut:

1. Mengembangkan *library* Flutter berbasis *code generation* untuk otomatisasi integrasi API dari dokumen Swagger/OpenAPI.
2. Menghasilkan struktur kode yang sesuai dengan prinsip *Clean Architecture* serta mendukung *dependency injection*.
3. Mengetahui tingkat keberhasilan dan keandalan fungsionalitas *library* melalui pengujian *Black Box Testing.*
4. Menganalisis tingkat penerimaan pengguna terhadap *library* Swagen berdasarkan pendekatan *Technology Acceptance Model* (TAM).

1.4 Manfaat

Penelitian ini diharapkan dapat memberikan kontribusi dalam pengembangan ilmu pengetahuan di bidang rekayasa perangkat lunak, khususnya pada kajian pengembangan aplikasi *mobile* dan otomatisasi integrasi *Application Programming Interface* (API). Melalui pengembangan *library* Flutter berbasis Swagger/OpenAPI, penelitian ini dapat menjadi referensi akademis mengenai penerapan *code generation* dalam pengembangan perangkat lunak serta pemanfaatan spesifikasi API sebagai dasar otomatisasi proses integrasi data secara terstruktur. Dari sisi pengembangan teknologi, hasil penelitian ini diharapkan mampu menghasilkan sebuah *library* Flutter bernama Swagen yang dapat membantu pengembang dalam mengotomatisasi proses integrasi API, sehingga mengurangi pekerjaan manual, meminimalkan kesalahan pemetaan data, dan meningkatkan konsistensi struktur kode pada proyek Flutter berskala besar maupun kompleks. Selain itu, penelitian ini memberikan manfaat dari aspek metodologis melalui penerapan metode *Extreme Programming* (XP) yang menekankan proses iteratif, adaptif, dan berorientasi pada kualitas. Penerapan XP dalam penelitian ini dapat menjadi contoh implementasi *Software Development Life Cycle* (SDLC) yang responsif terhadap perubahan kebutuhan teknis, sehingga dapat menjadi acuan bagi penelitian maupun pengembangan *library* serupa di masa mendatang.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, dapat disimpulkan bahwa pengembangan library Flutter Swagen berhasil menjawab seluruh rumusan masalah yang diajukan. Pertama, pada aspek perancangan dan pembangunan sistem, Swagen dikembangkan menggunakan metode *Extreme Programming (XP)* melalui tahapan iteratif yang mencakup *planning*, *design*, *coding*, dan *testing*. Library ini mampu mengotomatisasi proses integrasi API berbasis Swagger/OpenAPI melalui pendekatan *code generation*, yang dimulai dari pembacaan spesifikasi OpenAPI (format JSON, YAML, maupun URL), proses *parsing* struktur API seperti *paths*, *schemas*, dan *components*, hingga menghasilkan kode secara otomatis. Hasil *code generation* mencakup komponen pada *domain layer* seperti *entity*, *repository interface*, dan *use case*, *data layer* seperti *model*, *data source*, dan *repository implementation*. Hal ini menunjukkan bahwa proses integrasi API yang sebelumnya dilakukan secara manual dapat diotomatisasi secara sistematis dan konsisten.

Kedua, dalam implementasi arsitektur sistem, Swagen telah menerapkan prinsip *Clean Architecture* dengan memisahkan struktur kode ke dalam tiga layer utama, yaitu *data*, *domain*, dan *presentation*. Selain itu, sistem dilengkapi dengan folder *core* untuk menangani komponen umum seperti *error handling* melalui *failure* dan *exception*. Pada iterasi lanjutan, Swagen berhasil mengembangkan fitur pembangkitan *presentation layer* yang mencakup *state* dan *state management*, serta menerapkan *dependency injection* melalui mekanisme *injector* untuk mengelola dependensi antar komponen. Implementasi ini juga didukung oleh fitur otomatisasi pembentukan struktur folder berbasis *feature-based structure*, sehingga menghasilkan struktur proyek yang modular, konsisten, dan mudah dikembangkan. Dengan demikian, Swagen tidak hanya menghasilkan kode secara otomatis, tetapi juga memastikan kualitas arsitektur yang sesuai dengan praktik pengembangan perangkat lunak modern.

Ketiga, berdasarkan hasil pengujian menggunakan metode *Black Box Testing*, seluruh fitur utama Swagen berjalan sesuai dengan kebutuhan dan spesifikasi yang telah ditentukan. Pengujian dilakukan melalui berbagai skenario, antara lain konversi spesifikasi OpenAPI dari file dan URL, validasi input yang tidak sesuai, pengujian konsistensi hasil *generate*, pengujian pada spesifikasi dengan banyak endpoint, serta pengujian efisiensi waktu proses. Hasil pengujian menunjukkan bahwa sistem mampu menghasilkan seluruh komponen kode, meliputi *domain*, *data*, *presentation*, *injector*, serta struktur folder *Clean Architecture* dengan benar

tanpa kesalahan, serta mampu menangani kondisi *error* secara tepat. Hal ini menunjukkan bahwa Swagen telah memenuhi kebutuhan fungsional sistem dan layak digunakan sebagai solusi otomatisasi integrasi API.

Keempat, berdasarkan evaluasi menggunakan *Technology Acceptance Model* (TAM), Swagen memperoleh tingkat penerimaan yang baik hingga sangat baik pada seluruh kelompok responden, baik secara keseluruhan, mahasiswa, maupun non-mahasiswa (*developer*). Secara umum, nilai rata-rata *Perceived Ease of Use* (PEOU) sebesar 4,13 (baik), *Perceived Usefulness* (PU) sebesar 4,35 (sangat baik), *Attitude Toward Using* (ATU) sebesar 4,23 (sangat baik), *Behavioral Intention to Use* (BIU) sebesar 4,28 (sangat baik), serta *Actual Use* (AU) sebesar 3,67 (baik). Analisis lebih lanjut menunjukkan bahwa responden mahasiswa menilai Swagen mudah digunakan dan bermanfaat sebagai alat pembelajaran, sementara responden non-mahasiswa atau *developer* memberikan penilaian yang lebih tinggi, terutama pada aspek manfaat dan minat penggunaan, yang menunjukkan relevansi Swagen dalam konteks profesional. Meskipun nilai *Actual Use* masih berada pada kategori baik, hal ini menunjukkan bahwa penggunaan aktual masih dapat ditingkatkan seiring dengan pengembangan fitur lebih lanjut.

Secara keseluruhan, Swagen terbukti mampu meningkatkan efisiensi, kualitas, dan kecepatan pengembangan aplikasi Flutter berbasis API, serta memberikan pengalaman penggunaan yang baik dan konsisten bagi berbagai kalangan pengembang.

5.2 Saran

Berdasarkan hasil penelitian, pengembangan Swagen masih dapat ditingkatkan pada beberapa aspek. Pengembangan selanjutnya dapat menambahkan dukungan terhadap fitur OpenAPI yang lebih kompleks, seperti pengelolaan *authentication* yang lebih beragam. Selain itu, perlu dikembangkan fleksibilitas dalam metode komunikasi jaringan dengan mendukung berbagai *HTTP client* seperti Dio selain http, serta menyediakan opsi integrasi dengan berbagai *state management* seperti Bloc, Riverpod, atau lainnya. Pengembang juga dapat menambahkan dukungan terhadap JSON annotation dan library seperti *freezed* untuk otomatisasi pembuatan model yang lebih terstruktur.

Selain itu, Swagen juga berpotensi untuk dikembangkan lebih lanjut oleh mahasiswa sebagai bagian dari penelitian maupun proyek pengembangan perangkat lunak. Mahasiswa dapat melakukan pengembangan fitur tambahan, peningkatan performa, maupun eksplorasi integrasi dengan teknologi terbaru dalam ekosistem Flutter. Dengan demikian, Swagen tidak hanya berfungsi sebagai alat bantu pengembangan, tetapi juga dapat menjadi media pembelajaran dan

inovasi bagi mahasiswa dalam memahami konsep integrasi API, *code generation*, serta penerapan Clean Architecture secara lebih mendalam.

DAFTAR PUSTAKA

[1] Agil Maulana Nanda Riady, Paniran Paniran, and I Made Budi Suksmadana, "Perancangan Backend Api Berbasis Rest-Api pada Aplikasi Rekomendasi Resep Makanan," *Mars : Jurnal Teknik Mesin, Industri, Elektro Dan Ilmu Komputer*, vol. 2, no. 3, pp. 94–106, Jun. 2024, doi: 10.61132/mars.v2i3.137.

[2] C. Igwe-Nmaju, "Organizational Communication in the Age of APIs: Integrating Data Streams Across Departments for Unified Messaging and Decision-Making," *International Journal of Research Publication and Reviews*, vol. 5, no. 12, pp. 2792–2809, 2024, [Online]. Available: www.ijrpr.com

[3] M. Pandey and T. Dong, "Blending Accessibility in UI Framework Documentation to Build Awareness," in *ASSETS 2023 - Proceedings of the 25th International ACM SIGACCESS Conference on Computers and Accessibility*, Association for Computing Machinery, Inc, Oct. 2023. doi: 10.1145/3597638.3608380.

[4] S. A. Kinari, N. Funabiki, S. T. Aung, K. H. Wai, M. Mentari, and P. Puspitaningayu, "An Independent Learning System for Flutter Cross-Platform Mobile Programming with Code Modification Problems," *Information (Switzerland)*, vol. 15, no. 10, Oct. 2024, doi: 10.3390/info15100614.

[5] A. Md Sattar *et al.*, "Open Source Developments Accelerating Cross-platform Development with Flutter Framework," *Journal of Open Source Developments*, vol. 10, no. 2, 2023, doi: 10.37591/JoOSD.

[6] M. Liu, J. Wang, T. Lin, Q. Ma, Z. Fang, and Y. Wu, "An Empirical Study of the Code Generation of Safety-Critical Software Using LLMs," *Applied Sciences (Switzerland)*, vol. 14, no. 3, Feb. 2024, doi: 10.3390/app14031046.

[7] O. Baniaș, D. Florea, R. Gyalai, and D. I. Curiac, "Automated specification-based testing of REST APIs," *Sensors*, vol. 21, no. 16, Aug. 2021, doi: 10.3390/s21165375.

[8] I. P. A. Dharmaadi, E. Athanasopoulos, and F. Turkmen, "Fuzzing frameworks for server-side web applications: a survey," *Int. J. Inf. Secur.*, vol. 24, no. 2, Apr. 2025, doi: 10.1007/s10207-024-00979-w.

[9] S. Malvik, *Mastering azure API management: A practical approach to designing and implementing an API-centric enterprise architecture*. Apress Media LLC, 2022. doi: 10.1007/978-1-4842-8011-9.

[10] P. Palanisamy, "AI-Based Test Case Generation from Jira Stories, Swagger, or Code," *International Journal of Computer Techniques*, vol. 11, no. 3, 2024, [Online]. Available: https://ijctjournal.org/

[11] A. Fachrurozi, Lady Agustine, U. Faddillah, and I. Sugiyarto, "Implementasi Extreme Programming pada Pembuatan Website Sistem Informasi E-Accountant PT Naga Emas Internasional," *Remik: Riset dan E-Jurnal Manajemen Informatika Komputer*, vol. 9, no. 1, pp. 73–87, Jan. 2025, doi: 10.33395/remik.v9i1.14294.

[12] R. Juniar *et al.*, "E-Commerce Website Design Using Extreme Programming Method (Case Study: Umkm Bolu Ara)," 2025.

[13] F. Qazi, "Application Programming Interface (API) Security in Cloud Applications," *EAI Endorsed Transactions on Cloud Systems*, vol. 7, no. 23, p. e1, Oct. 2023, doi: 10.4108/eetcs.v7i23.3011.

[14] Y. Peng *et al.*, "Revisiting, Benchmarking and Exploring API Recommendation: How Far Are We?," *Institute of Electrical and Electronics Engineers Inc.*, vol. 49, no. 4, Dec. 2021, [Online]. Available: http://arxiv.org/abs/2112.12653

[15] S. Addanki, "Architectural Shifts in API Design: The Progression from SOAP to REST and GraphQL," *International Journal on Science and Technology (IJSAT)*, vol. 16, no. 2, 2025.

[16] S. Di Meglio, L. Libero, L. Starace, and S. Di Martino, "Starting a New REST API project? A Performance Benchmark of Frameworks and Execution Environments," in *CEUR Workshop Proceedings*, 2023. [Online]. Available: https://luistar.github.io/

[17] C. Iadanza, A. Trigila, P. Starace, A. Dragoni, T. Biondo, and M. Roccisano, "IdroGEO: A collaborative web mapping application based on REST API services and open data on landslides and floods in Italy," *ISPRS Int. J. Geoinf.*, vol. 10, no. 2, Feb. 2021, doi: 10.3390/ijgi10020089.

[18] A. Tuyishime, F. Basciani, J. L. C. Izquierdo, and L. Iovino, "Dynamic Provisioning of REST APIs for Model Management," in *Proceedings of the IEEE International Conference on Web Services, ICWS*, Jun. 2024, pp. 1335–1337. [Online]. Available: http://arxiv.org/abs/2406.17176

[19] O. Tkachenko, A. Chechet, M. Chernykh, S. Bunas, and P. Jatkiewicz, "Scalable Front-End Architecture: Building for Growth and Sustainability," *Informatica (Slovenia)*, vol. 49, no. 1, pp. 137–150, 2025, doi: 10.31449/inf.v49i1.6304.

[20] R. Jin, R. Cordingly, D. Zhao, and W. Lloyd, “GraphQL vs. REST: Investigating Performance and Scalability for Serverless Data Persistence,” in *IEEE International Conference on Cloud Engineering*, 2025, pp. 155–161.

[21] S. Suryawanshi, “Securing the Modern Web: A Comprehensive Exploration of Web API Authentication and Future Trends,” *International Journal of Trend in Scientific Research and Development (IJTSRD)* , vol. 9, no. 1, pp. 1026–1029, 2025, [Online]. Available: http://creativecommons.org/licenses/by/4.0

[22] B. A. Adewusi, B. Iyanu Adekunle, S. Damilola Mustapha, and A. C. Uzoka, “Advances in API-Centric Digital Ecosystems for Accelerating Innovation Across B2B and B2C Product Platforms,” *IRE Journals*, vol. 5, no. 3, 2021.

[23] A. D. Utmawati and A. Anggara, “Integrasi REST API Pada Aplikasi Mobile Untuk Monitoring Kerja Karyawan,” *JURNAL INFORMATIKA TEKNOLOGI DAN SAINS (JINTEKS)*, vol. 7, no. 4, pp. 1885–1891, 2025, doi: 10.51401/jinteks.v7i4.6852.

[24] L. da R. Araujo, G. S. V. Rodríguez, C. Marcos, and R. P. dos Santos, “EMPIRICAL ANALYSIS ON OPENAPI TOPIC EXPLORATION AND DISCOVERY TO SUPPORT THE DEVELOPER COMMUNITY,” *Computing and Informatics*, vol. 40, pp. 1345–1369, 2021, doi: 10.31577/cai.

[25] S. Orlov, T. Trunova, E. Hadzhyiev, H. Okhotko, Y. Bondar, and Y. Netrebin, “Usage of OpenAPI specification in distributed microservices-oriented architecture of the information system for astronomical data processing,” in *CEUR Workshop Proceedings*, 2024, pp. 532–544.

[26] SmartBear Software, “Swagger.”

[27] R. Kundra *et al.*, “OncoTree: A Cancer Classification System for Precision Oncology,” *JCO Clin Cancer Inform*, vol. 5, pp. 221–230, 2021, doi: 10.1200/CCI.20.

[28] R. Hutcheson *et al.*, “Software Architecture Reconstruction for Microservice Systems Using Static Analysis via GraalVM Native Image,” in *IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 12–22. doi: 10.1109/SANER60148.2024.00008.

[29] S. Casas, D. Cruz, G. Vidal, and M. Constanzo, “Uses and applications of the OpenAPI/Swagger specification: a systematic mapping of the literature,” in *Computer Science Society (SCCC) International Conference Chilean*, IEEE, 2021, pp. 143–146. doi: 10.1109/SCCC54552.2021.9650408.

[30] S. Serbout, F. Di Lauro, and C. Pautasso, “Web APIs Structures and Data Models Analysis,” in *2022 IEEE 19th International Conference on Software Architecture*

Companion, 2022, pp. 84–91. [Online]. Available: https://bigqueryconnection.googleapis.

[31] O. Baniaș, D. Florea, R. Gyalai, and D. I. Curiac, "Automated Specification-Based Testing of REST APIs," *Sensors*, vol. 21, no. 16, Aug. 2021, doi: 10.3390/s21165375.

[32] A. Lercher, J. Glock, C. Macho, and M. Pinzger, "Microservice API Evolution in Practice: A Study on Strategies and Challenges," *Journal of Systems and Software*, vol. 215, Nov. 2023, doi: 10.1016/j.jss.2024.112110.

[33] A. Tzavaras, N. Mainas, and E. G. M. Petrakis, "Thing Ontologies for the Semantic Web of Things," in *13th International Conference on Information, Intelligence, Systems and Applications, IISA 2022*, 2022. [Online]. Available: https://www.w3.org/2019/wot/hypermedia

[34] S. R. Uplenchwar, "Review on Detail Information About Flutter Cross Platform," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 10, no. 1, pp. 1016–1022, Jan. 2022, doi: 10.22214/ijraset.2022.39977.

[35] S. T. Aung, N. Funabiki, L. H. Aung, S. A. Kinari, M. Mentari, and K. H. Wai, "A Study of Learning Environment for Initiating Flutter App Development Using Docker," *Information (Switzerland)*, vol. 15, no. 4, Apr. 2024, doi: 10.3390/info15040191.

[36] O. M. A. AL-Atraqchi, "A Proposed Model for Build a Secure Restful API to Connect between Server Side and Mobile Application Using Laravel Framework with Flutter Toolkits," *Cihan University-Erbil Scientific Journal*, vol. 6, no. 2, pp. 28–35, Aug. 2022, doi: 10.24086/cuesj.v6n2y2022.pp28-35.

[37] R. Wahyudi *et al.*, "RANCANG BANGUN APLIKASI INVENTORI UMKM BERBASIS FLUTTER DENGAN INTEGRASI SHARED PREFERENCES DAN REST API," 2025. doi: 10.36040/jati.v9i6.16381.

[38] A. Fau and S. Artikel, "Pelatihan Pengenalan Dasar Framework Flutter dalam Pembangunan Aplikasi Mobile," 2024.

[39] R. P. Nair and M. G. Thushara, "NL2Code: A Hybrid NLP and Model-Driven Framework for Automated Code Generation from Natural Language and UML," in *2025 IEEE International Students' Conference on Electrical, Electronics and Computer Science, SCEECS 2025*, Institute of Electrical and Electronics Engineers Inc., 2025. doi: 10.1109/SCEECS64059.2025.10940505.

[40] R. A. Wijayanto, R. R. Hajar, and P. Sejati, "Implementing Flutter Clean Architecture for Mobile Tourism Application Development," 2023.

[41] Fajar Pradana, Raziqa Izza Langundi, Djoko Pramono, and Nur Ida Iriani, "Comparative Analysis of MVVM and MVP Patterns Performance on Android Dashboard System," *Jurnal Nasional Teknik Elektro dan Teknologi Informasi*, vol. 14, no. 2, pp. 87–95, May 2025, doi: 10.22146/jnteti.v14i2.18985.

[42] Firmansyah Firdaus Anhar, Made Hanindia Prami Swari, and Firza Prima Aditiawan, "Analisis Perbandingan Implementasi Clean Architecture Menggunakan MVP, MVI, Dan MVVM Pada Pengembangan Aplikasi Android Native," *Jupiter: Publikasi Ilmu Keteknikan Industri, Teknik Elektro dan Informatika*, vol. 2, no. 2, pp. 181–191, Jan. 2024, doi: 10.61132/jupiter.v2i2.155.

[43] S. Y. Ameen and D. Y. Mohammed, "Developing Cross-Platform Library Using Flutter," *European Journal of Engineering and Technology Research*, vol. 7, no. 2, pp. 18–21, Mar. 2022, doi: 10.24018/ejeng.2022.7.2.2740.

[44] R. Febrian, U. Rizki, and P. Studi Pendidikan Teknologi Informasi, "Pengembangan Platform Edukasi Berbasis Web untuk Tes Minat dan Bakat Siswa SLTA Menggunakan Framework Django," 2025. doi: 10.30599/farm3t26.

[45] S. A. Purnama, "Sistem Informasi Cash Flow dengan Menggunakan Metode Extreme Programming," *Jurnal Accounting Information System (AIMS*, vol. 8, no. 2, pp. 129–141, 2025, doi: 10.32627.

[46] S. W. Ramdany, S. Aulia Kaidar, B. Aguchino, C. Amelia, A. Putri, and R. Anggie, "Penerapan UML Class Diagram dalam Perancangan Sistem Informasi Perpustakaan Berbasis Web," *Journal of Industrial and Engineering System*, vol. 5, no. 1, pp. 30–41, 2024.

[47] Siska Narulita, Ahmad Nugroho, and M. Zakki Abdillah, "Diagram Unified Modelling Language (UML) untuk Perancangan Sistem Informasi Manajemen Penelitian dan Pengabdian Masyarakat (SIMLITABMAS)," *Bridge : Jurnal publikasi Sistem Informasi dan Telekomunikasi*, vol. 2, no. 3, pp. 244–256, Aug. 2024, doi: 10.62951/bridge.v2i3.174.

[48] M. Aqil *et al.*, "PENERAPAN BLACK BOX TESTING UNTUK EVALUASI FUNGSIONALITAS WEBSITE MAGGOPLAST," 2025.

[49] I. S. Handayanto and I. Nuryasin, "Pengujian Blackbox Decision Table pada Sistem Aplikasi Mobile Sharing Story App," *Smart Comp: Jurnalnya Orang Pintar Komputer*, vol. 13, no. 2, Apr. 2024, doi: 10.30591/smartcomp.v13i2.6572.

[50] M. D. Andika *et al.*, “Implementasi Black Box Testing Dalam Pengujian Fungsionalitas Website Leo Gym,” *URNAL TEKNOLOGI INFORMASIDIGITAL (JTID)*, vol. 1, no. 2, 2025.

[51] Al Afif Abdurrahman, Arya Abdul Mughni, Aida Sucia, Muhammad Ghozali, M. Ridho, and Subhanjaya Angga Atmaja, “Pengujian Aplikasi Mobile Gopay Menggunakan Equivalen Partitioning Metode BlackBox Testing,” *Jurnal Pengabdian Masyarakat dan Riset Pendidikan*, vol. 3, no. 4, pp. 5903–5911, Jun. 2025, doi: 10.31004/jerkin.v3i4.1518.

[52] K. A. Ramadhan, A. Meiriza, N. R. Oktadini, P. Putra, and P. E. Sevtiyuni, “Penerapan Metode Technology Acceptance Model Untuk Mengetahui Tingkat Penerimaan Pengguna Aplikasi Vidio,” *Jurnal Teknologi Dan Sistem Informasi Bisnis*, vol. 6, no. 2, pp. 266–274, Apr. 2024, doi: 10.47233/jteksis.v6i2.1319.

[53] M. N. Huda, M. Burhan, A. Satibi, H. A. Pradita, A. Saifudin, and I. Kusyadi, “Implementasi Black Box Testing pada Aplikasi Sistem Kasir dengan Menggunakan Teknik Equivalence Partitions,” *Jurnal Teknologi Sistem Informasi dan Aplikasi*, vol. 5, no. 2, p. 120, May 2022, doi: 10.32493/jtsi.v5i2.17645.

[54] Prathamesh Mali, “Flutter Retrofit Tutorial: Clean Architecture-Based API Integration,” Medium.

[55] Yamen Abdulrahman, “Clean Architecture in Flutter | MVVM | BloC | Dio,” Medium.

[56] Himanshu Gandhi, “Dependency Injection with Injectable: Boosting Your App’s Performance & Scalability,” Medium.

[57] L. O’Brien, T. Kanij, and J. Grundy, “Assessing gender bias in the software used in computer science and software engineering education,” *Journal of Systems and Software*, vol. 219, Jan. 2025, doi: 10.1016/j.jss.2024.112225.