Analisis Pengaruh Pola Arsitektur *Model View View Model* (MVVM) terhadap Kinerja Aplikasi *Mobile* dengan Menerapkan *Application Programming Interface* (API) Covid 19

1st Vianka Tetiana
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
viankatetiana@students.telkomuniversity.ac.id

2nd Dana Sulistiyo Kusumo
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
danakusumo@telkomuniversity.ac.id

3rd Monterico Andrian
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
monterico@telkomuniversity.ac.id

Abstrak—Aplikasi mobile dinilai semakin populer dan paling diminati oleh pengguna karena penggunaannya fleksibeldan praktis. Kinerja termasuk salah satu elemen terpenting dalam mendukung keberhasilan sebuah aplikasi karena berkaitan dengan seberapa cepat sistem berjalan dan memuat data. Dalam meningkatkan kinerja sistem, diperlukan efisiensi terhadap nilai metrik kinerja pada aspek resource utilization, capacity dan time behavior. Pemilihan architecture pattern merupakan salah satu aspek yang dapat mempengaruhi baik atau buruknya kinerja aplikasi. Pada Studi kasus ini, penulis melakukan perbandingan performansi pada aplikasi mobile yang menerapkan architecture pattern *Model View ViewModel* (MVVM) dan *Model View Presenter* (MVP) dengan menerapkan *Application Programming Interface* (API) Covid 19 sebagai perantara bagi untuk memperoleh informasi seputar Covid-19 untuk mendapatkan pengukuran metrik *CPU usage*, *memory usage*, *execution time* dan aspek *load time* API *data* yang paling unggul. Didapatkan penerapan MVVM dapat melakukan efisiensi nilai metrik kinerja dengan rata-rata metrik *execution time* yang dinilai lebih singkat sebesar 6682,666667 ms dan juga unggul pada pengujian *load time data* dari API berdasarkan rata-rata waktu *fetching data* JSON dari API sebesar 1082,333333 ms.

Kata kunci—*application programming interface, architecture pattern, model view viewmodel, model view presenter*

Abstract—Mobile applications are considered increasingly popular and most in demand by users because of flexibility and simple. Performance is one of the most important elements in supporting the success of an application becauseit is related to how fast the system runs and loads data. In improving system performance, efficiency of performance metric values is needed in the aspects of resource utilization, capacity and time behavior. Selection of the architecture pattern is one aspect that can affect the good or bad performance of the application. In this case study, the author compares the performance of mobile application that applies the Model View ViewModel (MVVM) and Model View Presenter (MVP) architecture patterns by implementing the Covid 19 Application Programming Interface (API) as an intermediary for obtaining information about Covid-19 for get the most superior measurement of CPU usage, memory usage, execution time and load time aspects of API data metrics. The implementation of MVVM can perform efficiency of performance metric values with an average execution time metric which is considered shorter by 5499.5 ms and also excels in testing load time data from the API based on the average JSON data fetching time from the API of 302.5 ms.

Keywords—application programming interface, architecture pattern, model view viewmodel, model view presenter

I. PENDAHULUAN

A. Latar Belakang

Perkembangan Smartphone di era sistem informasi dan modern, masyarakat terbiasa bekerja dengan komputer [1]. Penggunaan aplikasi mobile juga menjadi sektor baru dan berkembang pesat karena fleksibilitas dan kepraktisan penggunaannya [1]. Menurut Alisara Hincheranan dan Wanchai Rivepiboon [5], pesatnya pertumbuhan aplikasi mobile membutuhkan kinerja sistem yang tinggi untuk dapat beradaptasi dengan perubahan lingkungan dan perkembangan teknologi. Kinerja adalah aspek terpenting untuk keberhasilan aplikasi Anda karena bergantung pada seberapa cepat sistem Anda berjalan dan seberapa cepat data dimuat [2] [3]. Standar ISO/IEC 25010 [4] menyatakan bahwa kinerja yang baik dalam aplikasi mobile membutuhkan efisiensi dalam elemen utilisasi CPU, utilisasi memori, dan kecepatan eksekusi program. Salah satu faktor yang mempengaruhi kinerja sistem perangkat lunak adalah penerapan *architecture pattern* [6].

Dalam mengembangkan aplikasi mobile dengan kinerja yang baik, pola arsitektur merupakan hal penting yang harus diperhatikan karena aplikasi memiliki karakteristik dan pendekatan yang berbeda-beda [6] [13]. Sehingga perlu adanya penelitian untuk membandingkan beberapa aspek dari arsitektur perangkat lunak untuk mengetahui kelebihan dan kekurangan masing-masing arsitektur [13].

Terdapat beberapa architecture pattern yang banyak muncul dalam pengembangan aplikasi *mobile* dan banyak digunakan dalam aplikasi berat GUI, diantaranya terdapat tiga arsitektur yaitu *Model View Controller* (MVC), *Model View Presenter* (MVP) dan *Model View ViewModel* (MVVM) [6]. Dari penelitian sebelumnya [7] menyatakan, penerapan design pattern dengan cara memilih architecture pattern yang sesuai dapat meningkatkan performansi aplikasi menjadi lebih baik berdasarkan penggunaan *CPU* dan penggunaan memory. Pada pengujian performansi yang telah dilakukan sebelumnya [6], MVVM dan MVP memiliki kelebihan pada aspek testability, modifiability, dan performance dibandingkan MVC. MVVM dan MVP jauh lebih unggul pada metrik memory usage dan execution time dibandingkan dengan MVC [6].

Arsitektur MVVM yang merupakan framework baru dari pembentukan arsitektur MVP, salah satu perbedaannya adalah jika MVVM memiliki *Databinding* yang berperan mengurangi kompleksitas kode. MVVM juga memiliki komponen *LiveData* yang terdapat pada *ViewModel* [8]. *LiveData* ini berguna untuk menghindari adanya crash yang menyebabkan perangkat lunak berhenti berfungsi dengan benar dan menyebabkan *force quit* yang disebabkan karena Activity yang berhenti [8].

Sehingga tujuan penelitian ini adalah menerapkan MVVM pada aplikasi yang sebelumnya telah menerapkan MVP dengan menggunakan Covid-19 API sebagai *media* pertukaran *data.* Lalu aplikasi akan diuji fungsionalitas dengan menggunakan metode *Black Box Testing* untuk menguji sistem apakah aplikasi berjalan lancar tanpa error ataupun crash. Selanjutnya akan dilakukan analisis menggunakan pendekatan Goal Question Metric untuk mengetahui pengaruh penerapan MVVM terhadap kinerja aplikasi.

B. Topik dan Batasannya

Topik dan batasan masalah pada penelitian ini adalah melakukan analisis pengaruh penerapan *architecture pattern* MVVM terhadap kinerja pada aplikasi *mobile*. Untuk pengujian fungsionalitas menggunakan metode *Black Box testing* dan untuk analisis metrik kinerja menggunakan metode Goal Question Metrik. Adapun batasan masalah pada penelitian ini, diantaranya pengujian kinerja hanya mencakup metrik *CPU usage, memory usage, execution time* dan *load time data* dari API dan dalam mengukur kinerja aplikasi dilakukan pada *emulator* yang ada di fitur Android Profiler yang ada di *platform* Android Studio. Selanjutnya batasan lainnya yaitu aplikasi menggunakan *dataset* JSON yang diambil dari COVID-19 API.

II. KAJIAN TEORI

A. Penelitian Terkait

Pada tahun 2016, Tian Lou [6] melakukan penelitian mengenai analisis pengaruh MVC, MVP, dan MVVM pada aspek *testability, modifiability,* dan *performance* dan dari penelitian tersebut disimpulkan MVP dan MVVM lebih unggul dibandingkan dengan MVC pada metrik *memory usage dan execution time*, serta *coupling level* yang lebih rendah. Selanjutnya juga terdapat penelitian yang dilakukan Luka Omrčen, dkk [9] yang mengimplementasi MVP dan MVVM pada sistem aplikasi *warehouse management* dengan membandingkan dua arsitektur tersebut dan disimpulkan bahwa Penerapan MVVM dan MVP menunjukkan bahwa, pada metrik *CPU usage* tidak terdapat perbedaan yang signifikan, namun untuk *memory usage,* MVVM memerlukan ruang *memory* lebih dibandingkan MVP.

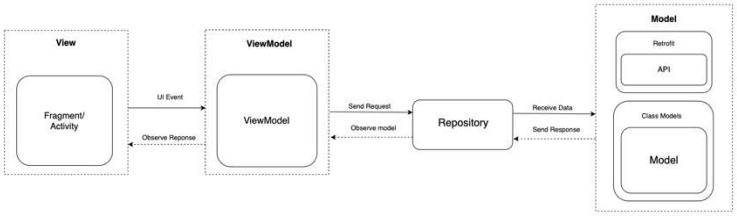
Dari penelitian diatas [6], dinyatakan bahwa MVVM dan MVP lebih unggul dibandingkan MVC. Serta didukung penelitian selanjutnya [9], untuk MVP dan MVVM unggul pada metriknya masing-masing. Sehingga pada penelitian ini, penulis akan membandingkan penerapan MVP dengan MVVM terhadap kinerja aplikasi *mobile*.

B. Performansi pada *Mobile Application*

Performansi pada suatu aplikasi diukur dari sejauh mana suatu sistem atau suatu komponen sistem dapat menyelesaikan fungsi yang ditentukan dalam batasan waktu yang ada, seperti kecepatan, akurasi atau penggunaan *memory* [6].

Disebutkan ISO/IEC 25010 [5] yang merupakan metode evaluasi yang digunakan pada penelitian yang fokus terhadap evaluasi beberapa aspek, salah satunya adalah *performance.* Untuk menghasilkan performansi yang baik pada aplikasi *mobile*, diperlukannya efisiensi terhadap *CPU Usage* yang merupakan pengukuran *basic metric* untuk memberikan *realtime performance* berdasarkan jumlah sumber daya yang digunakan pada aplikasi selama durasi skenario pengujian [5]. Selanjutnya efisiensi terhadap *memory usage* yang merupakan metrik pengukuran ruang yang diperlukan untuk mengeksekusi fungsi [5]. Dan yang terakhir terdapat pengukuran *execution time* yang merupakan metrik pengukuran waktu pemrosesan program saat kode dieksekusi pada *unit* pemrosesan pusat (CPU) [5].

C. *Model View ViewModel*



GAMBAR 1
SKEMA ARSITEKTUR MVVM

Pada Gambar 1 di atas ditunjukan bahwa, arsitektur MVVM atau *Model View ViewModel* terdiri dari *Model* yang berperan menyimpan *data* dan kode untuk memperbaiki dan memodifikasi *data* dan *Model* juga memperbarui *database* dengan berkomunikasi dengan *web server* atau API, *View* yang berperan menampilkan antarmuka pengguna tetapi tidak mengandung *data* yang ditampilkan kepada pengguna dan *ViewModel* berperan meneruskan *data* dari *Model* yang akan diteruskan ke *Activity* yang ada di komponen *View*, dan *View* berperan melakukan *binding data* yang berasal dari XML *layout* [6].

D. Black Box Testing

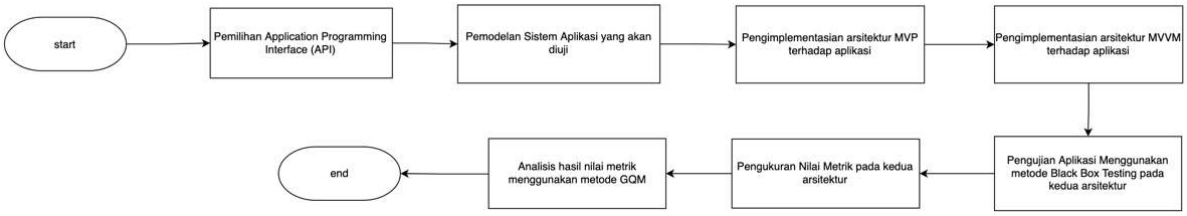
Pengujian untuk menguji *test case* pada *performance testing*, apakah aplikasi yang dirancang sesuai dengan spesifikasi kebutuhan persyaratan fungsional dan dapat berjalan tanpa adanya *error* ataupun *force close* [11]. Metode *Black box testing* fokus hanya pada itu *output* yang dihasilkan sesuai dengan kondisi input [11].

E. Goal Question Metric

Goal Question Metric atau GQM merupakan metode yang merepresentasukan struktur hierarkis yang mencakup 3 tingkat, diantaranya *conceptual level* yang mendefinisikan tujuan dari pengukuran, *operation level* yang mendefinisikan pertanyaan untuk mendapatkan jawaban yang akan dinilai dan *quantitative level* yang mendefinisikan data sebagai jawaban dari pertanyaan pertanyaan [12].

III. METODE

Berikut terdapat Ganbar 2 yang merupakan diagram tahapan pemodelan pada penelitian ini:



GAMBAR 2
DIAGRAM TAHAPAN PENELITIAN

A. *Application Programming Interface* (API)

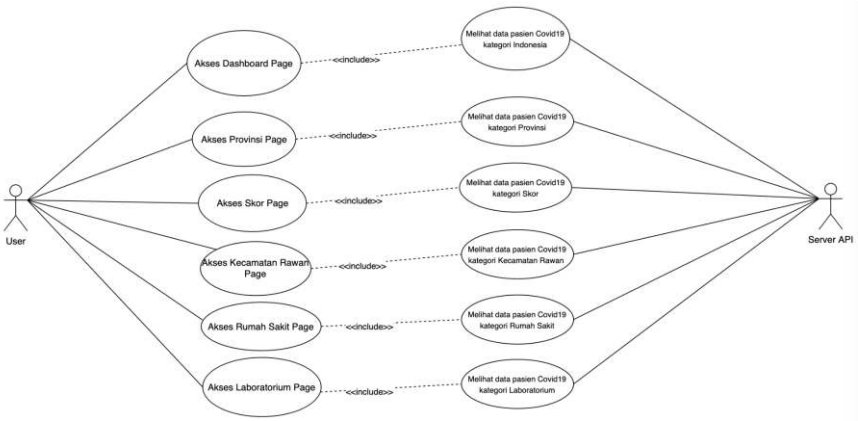
Pada penelitian ini data yang digunakan adalah *data* JSON yang berasal dari API sebagai perantara bagi aplikasi untuk memperoleh informasi seputar COVID-19 dari server secara fleksibel atau klien dan *server*. API yang digunakan adalah "Covid 19 API". Berikut terdapat Tabel 1 yang merupakan data realtime dari prov.json yang digunakan pada Provinsi page yang menampilkan update data pasien Covid-19 sesuai dengan kategorinya di setiap provinsi.

TABEL 1
DATA PROV.JSON

key	jumlah_kasus	jumlah_sembuh	Jumlah_meninggal	total_dirawat
DKI JAKARTA	1416206	1394170	15534	6502
JAWA BARAT	1175635	1150612	15952	9071
JAWA TENGAH	636824	602138	33510	1176
JAWA TIMUR	602642	570298	31780	564

B. Pemodelan Sistem Aplikasi untuk Pengujian

Aplikasi yang dirancang ditujukan untuk pengujian kinerja dengan menerapkan arsitektur MVP dan MVVM serta menerapkan API yang terdapat di *website* untuk menampilkan dan menyimpan data mengenai perkembangan kasus COVID-19 di Indonesia dan informasi lainnya seputar COVID-19. Untuk desain sistem aplikasi, berikut terdapat Gambar 3 dan Tabel 2. yang merupakan *use case diagram* dan *use case identification*:

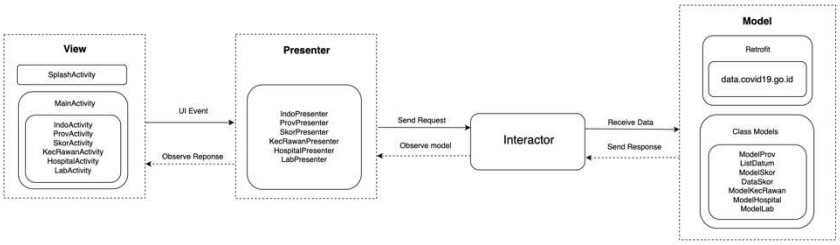


GAMBAR 3
USE CASE DIAGRAM

TABLE 2
USE CASE IDENTIFICATION

NO	USE CASE NAME	DESCRIPTION	ACTOR
1	AKSES DASHBOARD PAGE	*ACTOR* DAPAT MELIHAT HALAMAN UTAMA YANG BERISIKAN DATA PASIEN COVID -19 DI INDONESIA	*USER*
2	AKSES PROVINSI PAGE	*ACTOR* DAPAT MELIHAT TAMPILAN HALAMAN YANG BERISI INFORMASIMENGENAI NAMA PROVINSI DI INDONESIA	*USER*
3	AKSES SKOR PAGE	*ACTOR* DAPAT MELIHAT TAMPILAN HALAMAN YANG BERISI INFORMASIMENGENAI SKOR RESIKO PENULARAN COVID-19	*USER*
4	AKSES KECAMATAN RAWAN PAGE	*ACTOR* DAPAT MELIHAT TAMPILAN HALAMAN YANG BERISI INFORMASI MENGENAI NAMA KECAMATAN DI INDONESIA DAN KATEGORI RESIKO	*USER*
5	AKSES RUMAH SAKIT PAGE	*ACTOR* DAPAT MELIHAT TAMPILAN HALAMAN YANG BERISI INFORMASI MENGENAI DATA-DATA RUMAH SAKIT RUJUKAN PASIEN COVID-19	*USER*
6	AKSES LABORATORIUM PAGE	*ACTOR* DAPAT MELIHAT TAMPILAN HALAMAN YANG BERISI INFORMASI MENGENAI NAMA LABORATORIUM RUJUKAN PASIEN COVID-19	*USER*

C. Pengimplementasian Arsitektur MVP

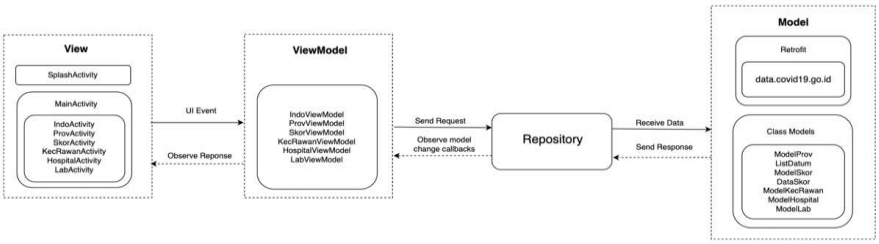


GAMBAR 4.
SKEMA ARSITEKTUR MVP PADA APLIKASI MOBILE

Terlihat pada Gambar 4 MVP terdiri dari tiga komponen, diantaranya Model, View dan Presenter. Untuk komponen View mencakup kelas Activity, terdapat satu Activity untuk setiap halaman, sehingga terdapat 8 Activity yang dapat dilihat pengguna, diantaranya yaitu SplashActivity, MainActivity, DashboardActivity, ProvActivity, SkorActivity, KecRawanActivity, HospitalActivity dan LabActivity. Selanjutnya untuk kategori Presenter terdapat enam kelas Presenter, diantaranya yaitu DashboardPresenter, ProvPresenter, SkorPresenter, KecRawanPresenter, HospitalPresenter dan LabPresenter, terdapat satu kelas Presenter untuk satu kelas View, sehingga dapat berisi beberapa Interactor yang masing-masing terdapat beberapa *procedure* atau *function* untuk mendapatkan data dari API dan kemudian memprosesnya sesuai dengan requirement yang ada dan meneruskannya ke View. Selanjutnya untuk kategori Model mencakup kelas-kelas model, diantaranya terdapat ModelHome, ModelProv, ListDatum, ModelSkor, DataSkor, ModelKecRawan, ModelHospital, ModelLab dan kelas RetrofitService yang akan menangani pengelolaan *data* dengan

library Retrofit.

D. Pengimplementasian Arsitektur MVVM



GAMBAR 5
SKEMA ARSITEKTUR MVVM PADA APLIKASI MOBILE

Dapat dilihat dari Gambar 5 Arsitektur MVVM terdiri dari komponen Model, View dan ViewModel. Untuk komponen View mencakup kelas *Activity*, terdapat satu Activity untuk setiap halaman, sehingga terdapat 8 Activity, diantaranya yaitu SplashActivity, DashboardActivity, ProvActivity, SkorActivity, KecRawanActivity, HospitalActivity dan LabActivity. Aplikasi menerapkan prinsip arsitektur MVVM, yaitu untuk satu View hanya memiliki satu ViewModel [1]. Selanjutnya untuk kategori ViewModel terdapat enam kelas ViewModel, diantaranya yaitu DashboardViewModel, ProvViewModel, SkorViewModel, KecRawanViewModel, HospitalViewModel dan LabViewModel, kelas-kelas ini mengimplementasikan *library* LiveData untuk menangani penyajian data yang akan ditampilkan di databinding untuk mencantumkan View dan ViewModel. Kemudian terdapat Repository yang digunakan oleh aplikasi untuk menangani koneksi ke API. Selanjutnya untuk kategori Model memiliki kesamaan seperti MVP, yaitu mencakup kelas-kelas model, diantaranya terdapat ModelHome, ModelProv, ListDatum, ModelSkor, DataSkor, ModelKecRawan, ModelHospital, ModelLab dan kelas RetrofitService yang akan menangani pengelolaan *data* dengan library Retrofit.

E. Black Box Testing

Metode *Black box testing* ditujukan untuk menguji fungsionalitas aplikasi yang menerapkan MVP dan MVVM berdasarkan beberapa *test case* yang ada. *Test case* dipilih berdasarkan *requirement* dari komponen Model,jenis data yang diambil dari API dan *respon size* dari API. Terdapat 3 *test case* ID, diantaranya TC-01, TC-02, TC-03 sebagai pengujian yang akan dilakukan untuk mengetahui kinerja aplikasi. Pendefinisian 3 *test case* ini didasari dengan alasan penulis melakukan pengujian terhadap *test case* ID TC-01 pada Dashboard page yang memiliki 1 Model kelas yang melakukan fetch data berupa list dari update.json yang berukuran 356.26 Kb. Selanjutnya penulis melakukan pengujian terhadap *test case* ID TC-02 pada Provinsi page yang memiliki 2 Model kelas yang merupakan inheritance Model yang melakukan fetch data berupa list dari update.json yang berukuran 26.52 Kb yang cukup memliki perbedaan size memory dengan data json yang diambil dari TC-01. Dan penulis melakukan pengujian terhadap *test case* ID TC-03 pada Kecamatan Rawan page yang memiliki 1 Model kelas yang melakukan fetch data berupa *arraylist* dari kecamatan_rawan.json yang berukuran 668.52 Kb. Berikut terdapat Tabel 3, Tabel 4, Tabel 5:

TABEL 3
TEST CASE 01

Identifier	TC-01 (Menampilkan Dashboard Page)
Related Requirement	Akses Dashboard Page
Description	Menampilkan halaman utama yang berisikan data pasien Covid -19 di Indonesia
Pre-condition	*User* mengklik aplikasi “CovidTracker”, setelah itu muncul *landing page*
Input data	*Click* aplikasi “CovidTracker”
Detailed steps	Ketika user melakukan click aplikasi “CovidTracker”, sistem akan menampilkan *splash screen* sebagai *landing page*. Selanjutnya sistem menampilkan dashboard page yang akan melakukan *fetch data* dari API update.json
Expected results	*Data* jumlah pasien terkonfirmasi, jumlah pasien meninggal, jumlah pasien sembuh, jumlah spesimen dan jumlah spesimen negatif dari update.json berhasil ditampilkan
Post-condition	*Data list* pasien Covid-19 di Indonesia sukses ditampilkan

TABEL 4
TEST CASE 02

Identifier	TC-02 (Menampilkan Provinsi Page)
Related Requirement	Akses Provinsi Page
Description	Menampilkan halaman Provinsi yang berisikan data pasien Covid -19 di setiap provinsi
Pre-condition	User mengklik aplikasi *option menu*, lalu user memilih opsi "Provinsi"
Input data	*Click option menu* pada *header* halaman
Detailed steps	Ketika user melakukan *click option menu* pada *header* halaman, sistem akan menampilkan beberapa opsi *menu* dan *user* melakukan *click* pada opsi "Provinsi". Selanjutnya sistem menampilkan Provinsi *page* yang akan melakukan *fetch data* dari API prov.json
Expected results	*Data* jumlah pasien terkonfirmasi, jumlah pasien meninggal, jumlah pasien sembuh, jumlah pasien dirawat dari update.json berhasil ditampilkan
Post-condition	*Data list* pasien Covid-19 di provinsi sukses ditampilkan

TABEL 5
TEST CASE 03

Identifier	TC-04 (Menampilkan Kecamatan Rawan Page)
Related Requirement	Akses Kecamatan Rawan Page
Description	Menampilkan halaman Kecamatan Rawan yang berisikan *data* kecamatan sebagai lokasi rawan penularan Covid-19
Pre-condition	*User* mengklik aplikasi *option menu*, lalu *user* memilih opsi "Kecamatan Rawan"
Input data	*Click option menu* pada *header* halaman
Detailed steps	Ketika user melakukan *click option menu* pada *header* halaman, sistem akan menampilkan beberapa opsi *menu* dan *user* melakukan *click* pada opsi "Rumah Sakit". Selanjutnya sistem menampilkan Kecamatan Rawan *page* yang akan melakukan *fetch data* dari API kecamatan_rawan.json
Expected results	*Data* nama kecamatan di Indonesia dan kategori resiko dari API kecamatan_rawan.json
Post-condition	*Data list* kecamatan sebagai lokasi rawan penularan Covid-19

F. Pengukuran Metrik Pengujian

Pada tahap pengukuran metrik pengujian ini ditujukan untuk mendapatkan variabel ukur, diantaranya *CPU usage*, memory usage, execution time dan load time API data.

G. Goal Question Metric

Pendekatan *Goal Question Metric* pada penelitian ini ditujukan untuk menganalisis pengaruh penerapan MVVM pada aplikasi *mobile* yang menerapkan Covid 19 API untuk mengambil *dataset*. Pendekatan GQM menjabarkan Goal untuk analisis perbandingan hasil pengujian kinerja aplikasi yang menerapkan MVP dan MVVM, Question atau pertanyaan untuk menanyakan apakah MVVM berpengaruh meningkatkan kinerja pada setiap metrik dan Metric merupakan objek dari tujuan dilakukannya pendekatan GQM. Berikut terdapat Tabel 6 yang menjabarkan tujuan utama dilakukannya pendekatan GQM pada tahap analisis.

TABEL 6
GQM UTAMA PENGUJIAN KINERJA

GQM		**Item**
Main Goal	*Purpose* *Issue* *Object* *Viewpoint*	Memastikan kinerja tinggi dari banyaknya metrik pengujian kinerja yang yang menerapkan MVVM pengguna

Question	Apakah pengaruh penerapan MVVM terhadap kinerja aplikasi
Metrics	Banyaknya metrik pengukuran kinerja yang menerapkan MVVM

IV. HASIL DAN PEMBAHASAN

A. *Black Box Testing*

Untuk pengujian fitur aplikasi yang sudah dibangun dan mengetahui aplikasi berfungsi baik dan tidak terjadi error dengan menggunakan metode *Black Box testing*. Berikut hasil pengujian fungsionalitas pada Tabel 7.

TABEL 7
HASIL PENGUJIAN FUNGSIONALITAS APLIKASI

Test Case	Status	
	MVP	MVVM
TC-01	Valid	Valid
TC-02	Valid	Valid
TC-03	Valid	Valid

B. Pengujian *CPU Usage*

Pengujian penggunaan CPU diukur berdasarkan jumlah persentase pemrosesan CPU tertinggi pada setiap *test case*yang menerapkan MVP dan MVVM. Berikut terdapat Gambar 6 yang merupakan bukti screen capture pengukuran *CPU usage* yang menerapkan MVVM serta terdapat Tabel 8 yang merupakan keseluruhan hasil pengukuran *CPU usage* yang menerapkan MVP dan MVVM:

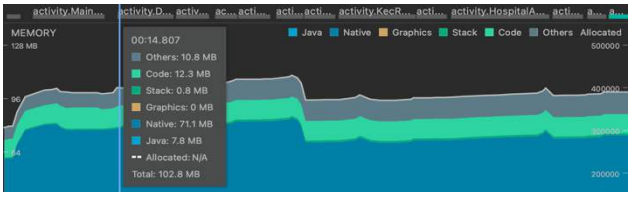
GAMBAR 6
HASIL PENGUJIAN *CPU USAGE* YANG MENERAPKAN MVVM PADA TC-01

TABEL 8
HASIL PENGUJIAN *CPU USAGE* PADA MVP DAN MVVM

Test Case	MVP	MVVM
TC-01	35	33
TC-02	31	34
TC-03	25	28

C. Pengujian *Memory Usage*

Pengujian penggunaan *memory* diukur berdasarkan jumlah penggunaan *memory* tertinggi pada setiap *test case* yang menerapkan MVP dan MVVM dan merupakan gabungan dari penjumlahan *Code, Stack, Graphics, Native* dan *Java.* Berikut terdapat Gambar 7 yang merupakan bukti screen capture pengukuran *memory usage* yang menerapkan MVVM serta terdapat Tabel 9 yang merupakan keseluruhan hasil pengukuran *memory usage* yang menerapkan MVP dan MVVM:



GAMBAR 7

HASIL PENGUJIAN *MEMORY USAGE* YANG MENERAPKAN MVVM PADA TC-01

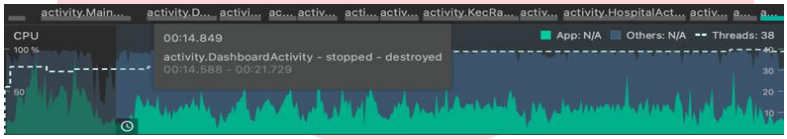
TC-03	75,9	83,3

TABEL 9
HASIL PENGUJIAN *MEMORY USAGE* PADA MVP DAN MVVM

Test Case	MVP	MVVM
TC-01	87,2	92
TC-02	89,8	93,5

D. Pengujian *Execution Time*

Pengujian waktu program dieksekusi diukur berdasarkan durasi dari *test case* saat mulai berjalan sampai berhenti. Berikut terdapat Gambar 8 yang merupakan bukti screen capture pengukuran *execution time* yang menerapkan MVVM serta terdapat Tabel 10 yang merupakan keseluruhan hasil pengukuran *execution time*yang menerapkan MVP dan MVVM:



GAMBAR 8
HASIL PENGUJIAN *EXECUTION TIME* YANG MENERAPKAN MVVM PADA TC-01

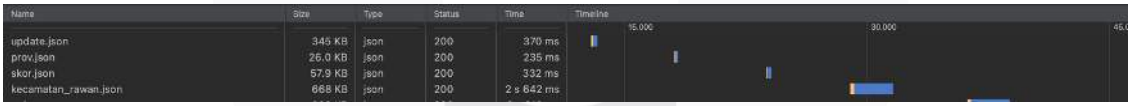
TC-03	6812	9049

TABEL 10
HASIL PENGUJIAN *EXECUTION TIME* PADA MVP DAN MVVM

Test Case	MVP	MVVM
TC-01	24990	8045
TC-02	1197	2954

E. Pengujian *Load Time API Data*

Pengujian waktu program saat memuat *data* JSON yang dibutuhkan dan berasal dari API. Berikut terdapat Gambar 9 yang merupakan bukti screen capture pengukuran *load time API data* yang menerapkan MVVM dan Tabel 11 yang merupakan keseluruhan hasil pengukuran *load time* API *data* untuk semua *test case* yang menerapkan MVP dan MVVM:



GAMBAR 9
HASIL PENGUJIAN *LOAD TIME* API *DATA* YANG MENERAPKAN MVVM PADA SEMUA *TEST CASE*

TABEL 11
HASIL PENGUJIAN LOAD TIME DATA API PADA MVP DAN MVVM

Test Case	JSON	MVP	MVVM
TC-01	update.json	8161	370
TC-02	prov.json	706	235
TC-03	kecamatan_rawan.json	3280	2642

F. *Goal Question Metric*

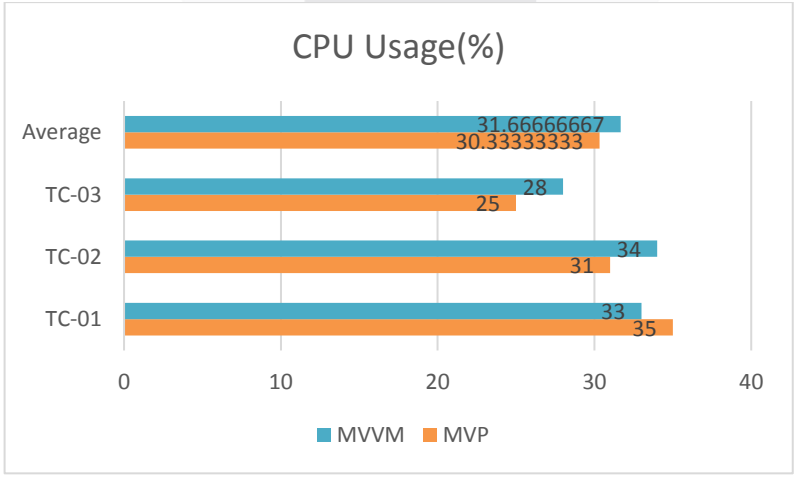
Sebelum melakukan analisis pengukuran kinerja yang telah dilakukan pada bagian 4.2, 4.3, 4.4 dan 4.5, penulis menerapkan pendekatan GQM untuk menjabarkan tujuan atau *Goal*, pertanyaan atau *Question* dan *Metrics* yang berkaitan dengan penelitian ini. Pada bagian 3.7 telah ditentukan *main goal* beserta *main question* dan *main metrics* pada pengujian kinerja ini. Berikut terdapat Tabel 12 yang merupakan daftar GQM pengujian kinerja:

TABEL 12
DAFTAR GQM PENGUJIAN KINERJA

GQM		**Item**
Goal 1	*Purpose* *Issue* *Object* *Viewpoint*	Memastikan kinerja tinggi dari hasil pengukuran *CPU usage* yang menerapkan MVP dan MVVM pengguna
Question		Apakah penerapan MVVM meningkatkan pengukuran metrik CPU usage?
Metrics		Rata-rata pengukuran *CPU usage* yang menerapkan MVP Rata-rata pengukuran *CPU usage* yang menerapkan MVVM
Goal 2	*Purpose* *Issue* *Object* *Viewpoint*	Memastikan kinerja tinggi dari hasil pengukuran *memory usage* yang menerapkan MVP dan MVVM pengguna
Question		Apakah penerapan MVVM meningkatkan pengukuran metrik *memory usage*?
Metrics		Rata-rata pengukuran *memory usage* yang menerapkan MVP Rata-rata pengukuran *memory usage* yang menerapkan MVVM
Goal 3	*Purpose* *Issue* *Object* *Viewpoint*	Memastikan kinerja tinggi dari hasil pengukuran *execution time* yang menerapkan MVP dan MVVM pengguna
Question		Apakah penerapan MVVM meningkatkan pengukuran metrik *execution time*?
Metrics		Rata-rata pengukuran *execution time* yang menerapkan MVP Rata-rata pengukuran *execution time* yang menerapkan MVVM
Goal 4	*Purpose* *Issue* *Object* *Viewpoint*	Memastikan kinerja tinggi dari hasil pengukuran *load time API data* yang menerapkan MVP dan MVVM pengguna
Question		Apakah penerapan MVVM meningkatkan pengukuran metrik *load time API data*?
Metrics		Rata-rata pengukuran *load time API data* yang menerapkan MVP Rata-rata pengukuran *load time API data* yang menerapkan MVVM

G. Evaluasi Nilai Metrik *CPU Usage*

Hasil dari pengukuran *CPU usage* yang ada pada goal 1 ditunjukkan pada Gambar 10 yang merupakan grafik perbandingan *CPU usage* untuk MVP dan MVVM:

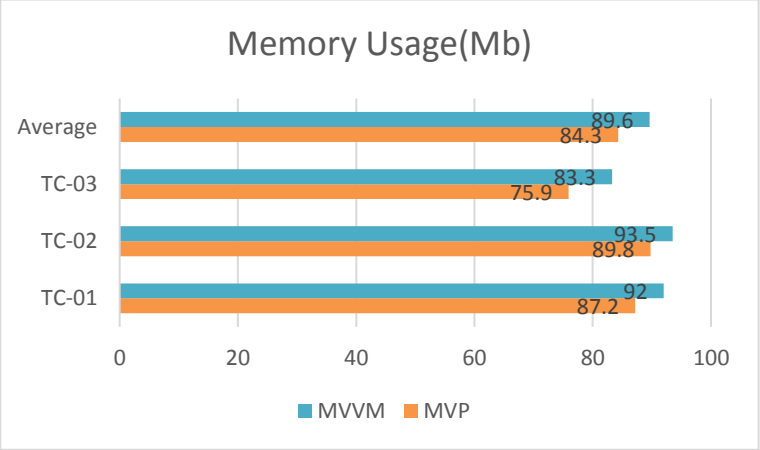


GAMBAR 10
GRAFIK PENGGUNAAN CPU UNTUK SEMUA *TEST CASE*

Semakin sedikit rata-rata penggunaan CPU, semakin berkurang beban CPU dan semakin efisien penggunaan CPU. Dari Gambar 5. diatas, dapat diketahui rata- rata untuk metrik CPU usage pada MVP adalah 30,5% dan untuk MVVM adalah 30,3333%, sehingga untuk kinerja CPU usage terbaik adalah MVVM karena penggunaan CPU lebih rendah dibandingkan MVP dengan kenaikan rata-rata 0,1667% pada nilai metrik CPU *usage* setelah menerapkan MVVM. Sehingga dari penerapan kedua ini, diketahui dengan menerapkan MVP mampun meningkatkan kinerja pada metrik penggunaan CPU.

H. Evaluasi Nilai Metrik *Memory Usage*

Hasil dari pengukuran *memory usage* yang ada pada goal 2 ditunjukkan pada Gambar 11 untuk MVP dan MVVM:

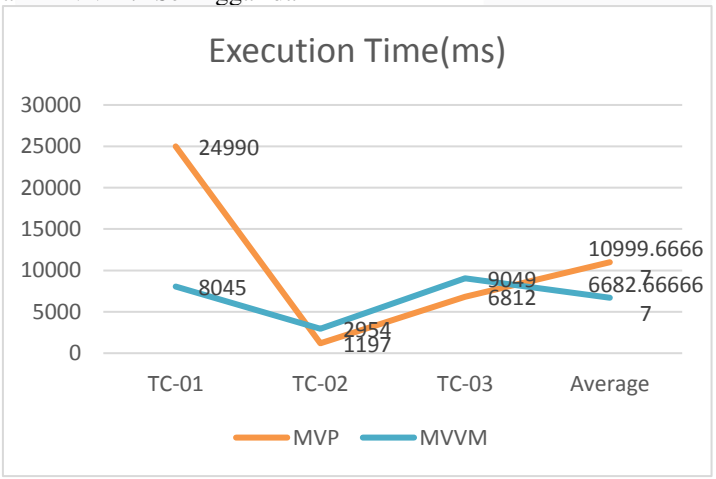


GAMBAR 11
GRAFIK PENGGUNAAN MEMORY UNTUK SEMUA *TEST CASE*

Semakin sedikit rata-rata penggunaan memory, semakin sedikit ruang penyimpanan yang dibutuhkan dan semakin efisien penggunaan memory. Dari Gambar 6. dapat diketahui rata-rata untuk metrik *memory usage* pada MVP adalah 89,6 Mb dan untuk MVVM adalah 84,3 Mb, sehingga untuk *kinerja memory usage* yang terbaik adalah penerapan MVP pada aplikasi dengan kenaikan rata-rata 5,3 Mb pada nilai metrik *memory usage* setelah menerapkan MVVM. Sehingga dari penerapan kedua ini, diketahui dengan menerapkan MVP mampu meningkatkan kinerja pada metrik penggunaan *memory*.

I. Evaluasi Nilai Metrik *Execution Time*

Hasil dari pengukuran *execution time* yang ada pada goal 3 ditunjukkan pada Gambar 12 untuk MVP dan MVVM:

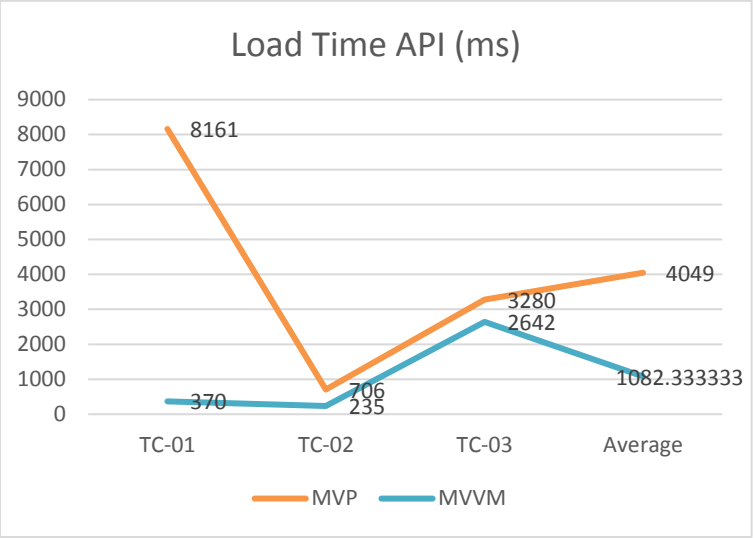


GAMBAR 12
GRAFIK WAKTU EKSEKUSI PROGRAM UNTUK SEMUA *TEST CASE*

Semakin sedikit rata-rata waktu eksekusi program, semakin cepat waktu program dieksekusi. Dari Gambar 12 diatas, dapat diketahui rata-rata untuk metrik *execution time* pada MVP adalah 10999,66667 ms dan untuk MVVM adalah 6682,666667 ms pada nilai metrik *execution time* data setelah menerapkan MVVM mengalami penurunan rata-rata durasi waktu 4317,000003 ms setelah menerapkan MVVM. Sehingga dari penerapan kedua ini, diketahui dengan menerapkan MVVM mampu meningkatkan kinerja pada metrik.

J. Evaluasi Nilai Metrik *Load Time API Data*

Hasil dari pengukuran load time data API yang ada pada goal 4 ditunjukkan pada Gambar 13 untuk MVP dan MVVM:



GAMBAR 13
GRAFIK *LOAD TIME* API *DATA* UNTUK SEMUA *TEST CASE*

Semakin sedikit rata-rata *load time* data API program, semakin cepat waktu *fetching data* JSON dari API. Dari Gambar 13 diatas, dapat diketahui rata-rata untuk aspek *load time API data* pada MVP adalah 4049 ms dan untuk MVVM adalah 1082,333333 ms, sehingga untuk kinerja *load time* API *data* terbaik adalah MVVM dengan penurunan rata-rata durasi waktu 2966,666667 ms pada nilai metrik load time API data setelah menerapkan MVVM. Sehingga dari penerapan kedua ini, diketahui dengan menerapkan MVVM mampu meningkatkan kinerja pada metrik load time data dari API.

K. Analisis Hasil Pengujian Nilai Metrik Kinerja yang Menerapkan MVVM

Berdasarkan evaluasi nilai metrik kinerja pada bagian 4.7, 4.8, 4.9 dan 4.10, rata-rata dari pengukuran nilai metrik CPU usage, memory usage, execution time dan load time API data untuk menjawab pertanyaan dari tujuan menggunakan metode GQM pada Goal 1, Goal 2, Goal 3 dan Goal 4. Berikut analisis rata-rata nilai metrik kinerja pada Tabel 13:

TABEL 13
ANALISIS RATA-RATA NILAI METRIK KINERJA

	Metrik Kinerja	
Analisis Rata-rata Nilai Metrik Kinerja	**Cpu Usage**	**Memory Usage**
	Rata-rata nilai metrik CPU usage yang sebelumnya menerapkan MVP mengalami peningkatan persentase penggunaan CPU setelah menerapkan MVVM, hal ini disebabkan dengan adanya class activity yang menerapkan data binding pada komponen View yang ada pada kelas Dashboard Activity dan Province Activity dan pada kelas XML, diantaranya pada kelas dashboard_row.xml dan province_row.xml. Sehingga dari penerapan DataBinding menyebabkan peningkatan penggunaan CPU, karena adanya proses mengintegrasi View dengan XML dengan objek data, setiap proses Binding akan melakukan satu pass pada View, proses ini dilakukan untuk mengekstrak View dengan ID. Dibandingkan dengan saat	Rata-rata nilai metriks memory usage yang sebelumnya menerapkan MVP mengalami peningkatan kebutuhan kapasitas memory setelah menerapkan MVVM, hal ini disebabkan karena adanya library tambahan yaitu LiveData yang terdapat di ViewModel berguna agar View dapat melakukan observasi data dari API. Selain itu terdapat library tambahan lainnya yaitu library DataBinding yang berfungsi untuk mengekstrak View dengan ID. Dibandingkan dengan saat menerapkan MVP, metrik memory usage tidak terdapat tambahan library LiveData maupun DataBinding.

	menerapkan MVP, metrik CPU usage cenderung lebih rendah karena hanya menerapkan findViewById dan setText karena tidak adanya annotation processors seperti DataBinding yang dapat meningkatkan kecepatan build time.	
	Execution Time	**Load Time API Data**
	Rata-rata nilai metrik execution time yang sebelumnya menerapkan MVP mengalami penurunan durasi waktu eksekusi program setelah menerapkan MVVM, hal ini disebabkan karena adanya penerapan DataBinding pada View, ViewModel dan XML untuk meringkas jumlah kode agar sistem tidak perlu melakukan inisialisasi layout sehingga dapat mempersingkat waktu eksekusi program. Dibandingkan dengan saat menerapkan MVP, metrik execution time usage cenderung lebih rendah karena hanya menerapkan findViewById dan setText, menyebabkan kecepatan proses eksekusi waktu cenderung lebih lambat.	Rata-rata nilai metrik load time API data mengalami penurunan durasi waktu saat fetch data dari API setelah menerapkan MVVM, hal ini disebabkan karena adanya penerapan LiveData yang berada di kelas ViewModel sehingga ViewModel dapat terhubung dengan Repository pada kelas RetrofitService untuk mendapatkan data dari API secara lebih cepat. Dibandingkan dengan saat menerapkan MVP, metrik load time API data saat fetch data lebih lambat karena pada method getData pada kelas ListData tidak menerapkan LiveData seperti yang ada di ViewModel pada MVVM.

Dari Tabel 17 di atas, dapat diketahui dengan menerapkan MVVM pada aplikasi mobile yang juga menerapkan Covid19 API yang menggunakan pendekatan GQM menunjukkan, adanya penerapan DataBinding dapat meningkatkan persentase nilai metrik CPU *usage* karena dapat meningkatkan kecepatan build time aplikasi, serta adanya kebutuhan kapasitas memory pada nilai metrik memory usage karena adanya library tambahan berupa DataBinding dan LiveData. Dari pernyataan tersebut menunjukkan dengan penerapan MVVM pada aplikasi mobile kurang tepat diterapkan jika ingin melakukan efisiensi terhadap metrik CPU usage dan memory usage. Namun penerapan MVVM pada aplikasi mobile tepat diterapkan jika ingin melakukan efisiensi terhadap metrik execution time dan load time API data, hal ini dikarenakan adanya penerapan DataBinding yang berfungsi untuk meringkas kode sehingga dapat mempersingkat waktu eksekusi program, serta adanya penerapan LiveData yang berfungsi mempercepat fetching data dari API.

V. KESIMPULAN

Dari penelitian analisis pengaruh pola arsitektur *Model View ViewModel* (MVVM) terhadap kinerja aplikasi *mobile* dengan menerapkan *Application Programming Interface* (API) Covid 19, dapat disimpulkan bahwa:

1. Penelitian dengan tujuan melakukan analisis terhadap aplikasi mobile yang telah menerapkan pola arsitektur MVVM sesuai dengan pemodelan yang diawali dengan pemilihan API dengan menggunakan Covid19 API lalu dilanjutkan dengan pemodelan sistem aplikasi yang akan diuji kinerjanya, setelah itu dilakukan pengimplementasian MVP dan MVVM terhadap aplikasi, lalu dilakukan pengujian fungsionalitas terhadap aplikasi, selanjutnya akan dilakukan pengukuran nilai metrik kinerja pada kedua arsitektur dan yang terakhir dilakukan analisis hasil nilai metrik kinerja menggunakan metode menunjukkan aplikasi dapat berjalan sesuai pemodelan.
2. Arsitektur MVVM da MVP unggul pada masing-masing 2 dari 4 metrik pengujian kinerja. MVVM unggul pada metrik *execution time* dan load time API data, sedangkan untuk MVP unggul pada metrik CPU usage dan memory usage. Untuk MVVM berdasarkan rata-rata waktu eksekusi sebesar 1050,8333 ms lebih cepat dibandingkan MVP sebesar 3132,3333 ms dengan penurunan durasi rata-rata waktu eksekusi sebesar 2081,5 ms. Selanjutnya MVVM juga unggul pada pengujian *load time data* dari API berdasarkan rata-rata waktu *fetching data* JSON dari API sebesar 6340,3333 ms, lebih cepat dibandingkan

MVP sebesar 7699 ms dengan penurunan durasi waktu fetch data sebesar 1328,6667 ms. Sedangkan untuk MVP unggul pada metrik CPU *usage* yang dinilai berdasarkan rata-rata penggunaan CPU sebesar 30,5%, persentasenya lebih sedikit penggunaannya dibandingkan MVVM sebesar 30,3333%, dengan penurunan penggunaan CPU sebesar 0,1667%. Lalu MVP juga unggul pada metrik penggunaan *memory* yang dinilai berdasarkan rata-rata penggunaan *memory* sebesar 83,183333 Mb lebih sedikit penggunaannya dibandingkan MVVM sebesar 89,216667 Mb dengan penurunan penggunaan *memory* sebesar 6,033334 Mb.

3. Dari analisis pengaruh pola arsitektur MVVM terhadap kinerja aplikasi mobile dengan menerapkan Covid 19 API yang dilakukan dengan pendekatan *Goal Question Metric* menunjukkan, MVVM lebih baik diterapkan jika ingin melakukan efisiensi terhadap nilai metrik *execution time* karena adanya penerapan DataBinding pada View, ViewModel dan XML untuk meringkas jumlah kode sehingga dapat mempersingkat waktu eksekusi program dan nilai metrik *load time* API *data* karena adanya penerapan LiveData yang berada di kelas ViewModel sehingga ViewModel dapat terhubung dengan Repository untuk mendapatkan data dari API secara lebih cepat.

Saran lanjutan dari penelitian ini yaitu dapat menerapkan *test case* yang representatif dan menambah jumlah *test case*, karena semakin banyak jumlah *test case* semakin baik.

REFERENSI

[1] Islam, Rashedul, Rofiqul Islam, and Tohidul Mazumder. "Mobile application and its global impact." International Journal of Engineering & Technology 10.6 (2010): 72-78.

[2] L. Corral and I. Fronza, “Better Code for Better Apps: A Study on Source Code Quality and Market Success of Android Applications,” May 2015. doi: 10.1109/MobileSoft.2015.10.

[3] L. Corral, A. Sillitti, and G. Succi, “Mobile multiplatform development: An experiment for performance analysis,” Procedia Comput. Sci., vol. 10, pp. 736–743, 2012, doi: 10.1016/j.procs.2012.06.094.

[4] ISO, “ISO/IEC 25010:2011,” 2011, [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en:en [Accessed 25 Januari 2022]

[5] Hincheeranan, Alisara, and Wanchai Rivepiboon. "A maintainability estimation model and tool." International Journal of Computer and Communication Engineering 1.2 (2012): 143.

[6] Lou, Tian. "A comparison of Android Native App Architecture MVC, MVP and MVVM." Eindhoven University of Technology (2016).

[7] Qasim, Awais, et al. “Evaluating the Impact of Design Pattern Usage on Energy Consumption of Applications for Mobile Platform.” Applied Computer Systems 26.1 (2021): 1-11.

[8] Maharjan, Bikesh. “Puzzle game using Android MVVM Architecture,” (2018).

[9] Omrčen, Luka, et al. "USPOREDBA ARHITEKTONSKIH UZORAKA MVVM I MVP U APLIKACIJI ANDROID-STUDIJA SLUČAJA SUSTAVA UPRAVLJANJA SKLADIŠTEM." ODRŽAVANJA' (2017): 97.

[10] Qureshi, M., and Fatima Sabir. "A comparison of model view controller and model view presenter." arXiv preprint arXiv:1408.5786 (2014).

[11] Nidhra, Srinivas, and Jagruthi Dondeti. "Black box and white box testing techniques-a literature review." International Journal of Embedded Systems and Applications (IJESA) 2.2 (2012): 29-50.

[12] Sim, Yee Wai, et al. "Goal Question Metric as an Interdisciplinary Tool for Assessing Mobile Learning Application." International Journal of Advanced Computer Science and Applications 13.5 (2022).

[13] Akhtar, Nayab, dan Sana Ghafoor. "Analisis Pola Arsitektur untuk Pengembangan Android." (2021).