Analisis Perbandingan Tingkat *Maintainability* antara Arsitektur MVVM (*Model-View-ViewModel*) dan MVC (*Model-View-Controller*) dalam Pengembangan Aplikasi *Mobile*

1st Maulana Sidiq
Telkom University
Fakultas Informatika
Bandung, Indonesia
maulanas@student.telkomuniversity.ac.id

2nd Dawam Dwi Jatmiko Suwawi
Telkom University
Fakultas Informatika
Bandung, Indonesia
dawamdjs@ telkomuniversity.ac.id

3rd Sri Widowati
Telkom University
Fakultas Informatika
Bandung, Indonesia
sriwidowati@ telkomuniversity.ac.id

***Abstrak* — Penelitian ini bertujuan untuk membandingkan nilai antara Model-View-ViewModel (MVVM) dan Model-View-Controller (MVC) dalam konteks pengembangan aplikasi mobile. Meningkatnya penggunaan aplikasi perangkat lunak berbasis mobile menjadi tantangan tersendiri bagi para pengembang dalam hal pemeliharaan aplikasi dalam jangka panjang. Oleh karena itu, sangat penting untuk memilih pola desain arsitektur yang sesuai untuk aplikasi perangkat lunak berbasis mobile. Penelitian ini akan memberikan panduan yang berharga bagi para pengembang untuk memilih arsitektur yang sesuai dengan kebutuhan pengembangan aplikasi perangkat lunak dalam jangka panjang.**

***Kata kunci*— Pengembangan Aplikasi Mobile, , Pola Desain Arsitektur, Model-View-ViewModel, Model-View-Controller, Pengembangan Aplikasi Perangkat Lunak**

I. PENDAHULUAN

Aplikasi perangkat lunak berbasis mobile sudah menjadi salah satu bagian yang penting pada kehidupan sehari-hari, hal ini didukung dengan perkembangan aplikasi perangkat lunak berbasis mobile yang mengalami peningkatan pada beberapa tahun terakhir. Penelitian terbaru menunjukkan bahwa pengguna aplikasi mobile semakin cepat berkembang di berbagai bidang, seperti e-commerce, pendidikan, dan lain sebagainya dalam kehidupan sehari-hari. Menurut data dari "Digital 2024: Indonesia" tercatat ada 353,3 juta perangkat mobile yang terhubung pada tahun 2024 di indonesia, yang setara dengan 126,58 persen dari total populasi warga indonesia dimana pada januari 2024 tercatat total ada 278,7 juta penduduk[1]

Dengan tingginya pengguna aplikasi perangkat lunak berbasis mobile, telah menciptakan tantangan baru bagi pengembang aplikasi perangkat lunak berbasis mobile terkait kemampuan maintainability dari aplikasi tersebut [2]. Maintainability (kemudahan pemeliharaan) merupakan salah satu faktor penting yang memengaruhi kualitas dan perkembangan aplikasi perangkat lunak berbasis mobile dalam jangka panjang [3]. Untuk menghadapi tantangan dalam pengembangan aplikasi perangkat lunak berbasis mobile, maka diperlukan pemilihan pola desain arsitektur yang sesuai bagi aplikasi perangkat lunak berbasis mobile tersebut [4].

Berdasarkan penelitian sebelumnya pada artikel "A Tale of Two Development Approach Empirical Study on The Maintainability and Modularity of Android Mobile Application with Anti-Pattern and Model-View-Presenter Design Pattern" memberikan sebuah rekomendasi penelitian lebih lanjut unutk membandingkan tingkat maintainability antara MVVM (Model-View-ViewModel) dan MVC (Model-View-Controller) dalam konteks pengembangan aplikasi mobile [2].

Dengan demikian pada penelitian ini akan berfokus pada perbandingan nilai maintainability antara (Model-View-ViewModel) dan MVC (Model-View-Controller) yang diharapkan dapat memberikan panduan yang berharga bagi pengembang dalam pemilihan arsitektur yang sesuai dengan kebutuhan perkembangan aplikasi perangkat lunak dalam jangka panjang.

II. KAJIAN TEORI

Menyajikan dan menjelaskan teori-teori yang berkaitan dengan variabel-variabel penelitian. Poin subjudul ditulis dalam abjad.

A. MVVM (Model VIiew ViewModel)

Model-View-ViewModel (MVVM) dikembangkan oleh Ken Cooper, dan John Gossman arsitek Microsoft. MVVM dikembangkan sebagai solusi dari kelemahan pola desain Model-View_Presenter (MVP) [5]. Model-View-ViewModel (MVVM) dikembangkan terdiri dari tiga komponen utama yaitu, Model sebagai data atau logika bisnis dalam aplikasi, View sebagai antar muka pengguna dalam aplikasi, dan ViewModel sebagai penghubung antara Model dan View [6].

MVVM memungkinkan pengembang untuk memisahkan logika bisnis dari antar muka pengguna, memudahkan pengujian, dan memungkinkan pengelolaan kode yang lebih

baik [7]. Penjelasan tiap komponen dapat dilihat pada figur II.1 [8].

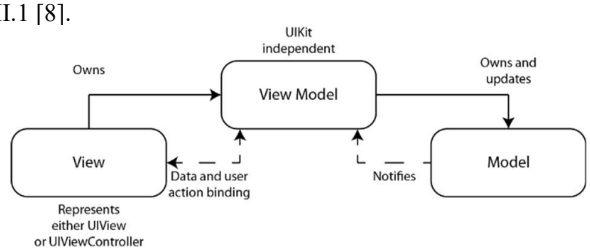

Figure 1 MVVM Pattern[9]

Pada figure 1 menunjukkan hubungan dan alur data antar komponen-komponen MVVM. Berikut adalah penjelasan singkatnya :

- Model merupakan komponen yang bertanggung jawab untuk merepresentasikan data dan aturan bisnis yang berjalan pada aplikasi.
- View merupakan komponen yang menampilkan tampilan pengguna dan bertanggung jawab untuk menampilkan data dari ViewModel kepada pengguna, dan menerima input dari pengguna.

ViewModel merupakan komponen yang bertanggung jawab untuk menyediakan data dan perilaku yang dibutuhkan oleh View, dan menangani interaksi pengguna pada View.

B. MVC (Model View Controller)

Model-View-Controller (MVC) dikembangkan dan di publikasikan berdasarkan rumusan ide dari salah satu peneliti Xerox PARC yaitu Reendkaug ketika bekerja di Xerox PARC pada masa 1970-an [9]. Pada saat ini Model-View-Controller (MVC) menjadi pola desain perangkat lunak yang umum digunakan untuk mengembangkan aplikasi, dimana logika pemrograman pada MVC dibagi menjadi 3 komponen terhubung. Komponen pada MVC yaitu Model sebagai representasi internal terkait informasi aplikasi, View sebagai antar muka pengguna pada aplikasi, dan Controller sebagai pengontrol logika pemrograman dalam aplikasi.

Untuk memahami bagaimana MVC bekerja, dapat dilihat pada diagram berikut [9]

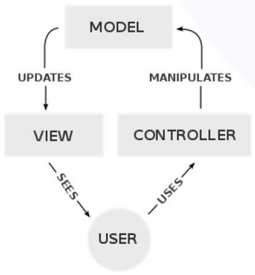

Figure 2 MVC Patter[10]

Pada figure 2 menunjukkan hubungan dan alur data antar komponen-komponen MVVM. Berikut adalah penjelasan singkatnya :

- View merupakan komponen yang menangani tampilan atau antarmuka pengguna dari aplikasi dan menampilkan data dari model untuk pengguna, serta menerima input dari pengguna.
- Model merupakan komponen untuk mengelola data aplikasi serta aturan bisnis yang terdapat pada aplikasi.

Controller berperan sebagai perantara antara Model dan View, dimana pada komponen ini akan menerima input pengguna dari View dan meneruskan ke Model untuk diproses

III. METODE

Berikut adalah alur metode penelitian yang akan dilakukan agar mencapai tujuan penelitian.

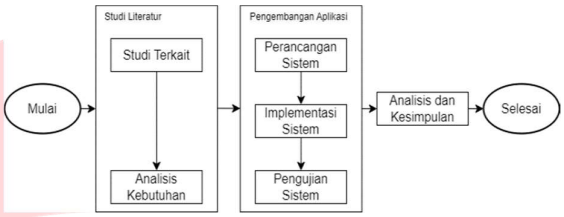

Figure 3 Methodologi Penelitian

Definisikan singkatan dan akronim saat pertama kali digunakan dalam teks, bahkan setelah didefinisikan dalam abstrak. Jangan mengggunakan singkatan dalam judul kecuali jika tidak dapat dihindari.

A. STUDI LITERATUR

Tahap ini dilaksanakan selama pengerjaan penelitian yang bertujuan untuk mengumpulkan dan menganalisis informasi yang relevan dengan topik penelitian, yaitu perbandingan pengaruh nilai maintainability pada program aplikasi mobile yang menerapkan dua arsitektur berbeda yaitu MVVM dan MVC. Studi literatur dapat mencakup sumber-sumber seperti buku, jurnal, artikel, tesis, laporan, dll. Studi literatur dapat membantu untuk mengetahui latar belakang masalah, tujuan penelitian, rumusan masalah, hipotesis, kerangka teori, metodologi, dan analisis data yang sesuai dengan kebutuhan pada topik penelitian ini.

B. PENGEMBANGAN APLIKASI

a. Perancangan sistem

Pada penelitian ini, perancangan sistem bertujuan untuk menentukan pola desain atau arsitektur pemrograman yang akan digunakan serta merancang antar muka, dan komponen-komponen dari aplikasi yang dikembangkan. Perancangan sistem dapat mencakup pemilihan kerangka kerja, use case diagram dan pembuatan prototipe yang akan digunakan pada penelitian ini.

Kerangka kerja yang digunakan pada penelitian ini adalah Flutter. Flutter merupakan sebuah platform pengembangan aplikasi mobile yang dibuat oleh Google. Flutter menggunakan bahasa pemrograman Dart, yang merupakan sebuah bahasa pemrograman yang bersifat objek, kelas, dan fungsional. Flutter memiliki keunggulan seperti:

- Dapat menghasilkan aplikasi yang berjalan di berbagai platform, seperti Android, iOS, web, dll.
- Dapat membuat aplikasi yang responsif, atraktif, dan kaya akan fitur visual dan animasi.

- Dapat melakukan hot reload dan hot restart, yang memungkinkan untuk melihat perubahan kode secara langsung tanpa harus melakukan kompilasi atau instalasi ulang.
- Dapat menggunakan widget, yang merupakan elemen dasar dari UI pada Flutter. Widget dapat dikombinasikan dan disesuaikan untuk membuat antarmuka yang sesuai dengan kebutuhan.

Pada penelitian ini sistem aplikasi yang digunakan sebagai objek penelitian yaitu aplikasi penjadwalan “MyTodo” yang merupakan aplikasi manajemen penjadwalan sebuah kegiatan atau sebagai pengingat, dan sistem aplikasi absensi "Sisfo" yang merupakan aplikasi absensi dan monitoring materi kegiatan belajar di salah satu pondok pesantren mahasiswa yang berada di bandung yaitu "PPMRJ(Pondok Pesantren Mahasiswa Roudhotul Jannah)".

Dari kedua aplikasi yang digunakan sebagai objek penelitian, penulis telah mendapatkan izin resmi/legal untuk aplikasi "Sisfo" dari pemilik aplikasi tersebut, yaitu PPMRJ (Pondok Pesantren Mahasiswa Roudhotul Jannah), terutama terkait perlindungan data pribadi pengguna aplikasi tersebut. Sementara itu, aplikasi "MyTodo" merupakan aplikasi yang dikembangkan sendiri oleh penulis sebagai objek penelitian, dan data yang digunakan dalam aplikasi ini berupa local database yang juga dikembangkan oleh penulis, sehingga tidak memerlukan perizinan.

Adapun Use Case diagram dari perancangan sistem aplikasi ini sebagai berikut :

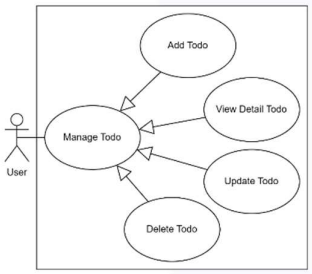

Figure 4 UseCase "MyTodo"

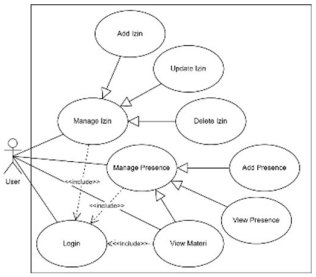

Figure 5 UseCase "Sisfo"

b. implementasi sistem

Berdasarkan use case pada perancangan sistem. kedua aplikasi ini diimplementasikan dan dikembangkan sebanyak dua kali dengan pola desain berbeda yaitu MVVM dan MVC. Keduanya dikembangkan secara berurutan dengan pengembangan pertama menggunakan pola desain MVVM dan selanjutnya dikembangkan dengan MVC. Kode antar muka antara kedua basis kode dirancang memiliki jumlah komponen yang sama dengan tujuan agar yang mempengaruhi hasil pengujian adalah dari pola desain yang diimplementasikan.

Pada penelitian ini penggunaan dua sistem aplikasi sebagai objek penelitian didasari dari tingkat keberhasilan dan kendala implementasi dari kedua sistem aplikasi tersebut. Dimana tingkat keberhasilan implementasi dalam persen dari kedua sistem aplikasi ini adalah 100% untuk “MyTodo” dan 33% untuk “Sisfo”.

Kendala implementasi pada sistem aplikasi “Sisfo” yang menyebabkan presentase keberhasilan implementasi pada sistem aplikasi ini hanya 33% adalah terjadi karena pemrograman dari sistem aplikasi sisfo yang sudah ada yaitu berbasis web memiliki struktur yang tidak tertata. Dimana pemrograman frontend dan database pada sistem tersebut menyatu dalam satu komponen dan tidak terpisah, sehingga dalam proses pengembangan "Sisfo" berbasis mobile terkendala pengambilan database yang tidak dapat langsung menggunakan APIs, serta keterbatasan waktu implementasi dalam pengembangan sistem aplikasi ini sehingga dari total 9 fitur yang direncanakan pada perancangan sistem hanya 3 fitur yang berhasil di implementasikan.

Oleh karena itu, sistem aplikasi “MyTodo” hadir sebagai sistem aplikasi pengganti serta pendukung sebagai objek penelitian pada penelitian ini.

Berikut adalah detail kedua aplikasi dalam proses implementasi sistem aplikasi:

Tabel 1 Detail Implementasi

Sistem Aplikasi		***Sisfo***	***MyTodo***
Kerangka Kerja		Flutter	Flutter
Arsitektur		MVVM MVC	MVVM MVC
Database		MySql	Local Database
total LOC(Line of Code)	***MVVM***	1579 Lines	457 Lines
	MVC	1652 Lines	456 Lines
total Number of Line in IDE	***MVVM***	2511 Line	690 Lines
	MVC	2486 Lines	695 Lines
Total komponen atau file (.dart)		30 Files	10 Files
Keberhasilan Implementasi (%)		33%	100%

Fitur	Add Izin All Presence Detail Presence Home Page Login Lorong Monitoring Profile Report User	DashBoard Todo List Detail Todo Add Todo Update Todo Delete Todo

c. pengujian sistem

Setelah dilakukan tahap perancangan dan implementasi pada sistem aplikasi, selanjut mulai dilakukan pengujian terhadap sistem yang sudah di bangun. Pengujian pada penelitian ini yaitu pengujian maintainability indeks pada sebuah program aplikasi yang sudah di bangun.

Persamaan Maintainability Index (MI) adalah sebuah metrik pengukuran yang menunjukkan seberapa mudah atau sulit untuk melakukan pemeliharaan sebuah program [10]. MI dapat dihitung dengan menggunakan rumus berikut[10] :

$$MI = 171 - 5.2\ln(HV) - 0.23CC - 16.2\ln(LOC) + (50 \times \sin(\sqrt{2.46 \times perCM})) \quad (1)$$

Dimana :

- HV (Halstead Volume), yaitu ukuran yang menunjukkan jumlah informasi yang terkandung dalam program. HV dapat dihitung dengan menggunakan rumus berikut :

  $$HV = (N_1 + N_2) \times \log_2(n_1 + n_2) \quad (2)$$

  Dimana :

  - N1 adalah jumlah total operator dalam program
  - N2 adalah jumlah total operand dalam program
  - n1 adalah jumlah total operator unik dalam program
  - n2 adalah jumlah total operand unik dalam program

- CC (Cyclomatic Complexity), yaitu ukuran yang menunjukkan jumlah cabang atau alur yang mengikuti dalam progra,. CC dapat dihitung dengan rumus berikut :

  $$CC = E - N + 2P \quad (3)$$

  Dimana :

  - E adalah jumlah edge (garis) dalam graft alur program.
  - N adalah jumlah node (titik) delam graft alur program.
  - P adalah jumlah komponen terhubung dalam graft alur program.

- LOC (Lines of Code), yaitu jumlah baris kode dalam program.

- perCM (Percent line of comment), yaitu persentase jumlah koment yang terdapat dalam program. perCM dapat dihitung dengan rumus:

  $$perCM = \frac{jumla\ \ baris\ komentar}{total\ baris\ kode} \times 100 \quad (4)$$

Hasil kalkulasi akan menghasilkan nilai MI, dimana nilai yang lebih tinggi menunjukkan maintainability yang baik. Berikut adalah kategori nilai MI :

Tabel 2 Index Maintainability

Nilai Maintanability Index	Klasifikasi
$MI > 85$	Highly maintainable
$65 < MI \leq 85$	Moderately maintainable
$MI \leq 65$	Difficult to maintain

Pada proses pengujian *maintainability index* dalam penelitian ini memanfaat kan alat bantu pengujian *maintainability* yaitu "SonarQube" yang sudah di setup untuk menganalisis sistem aplikasi dengan kerangka kerja flutter[11]. Pada "SonarQube" dapat melakukan analisis berdasarkan setiap metrik komponen penyusun rumus MI (*Maintainability Index*), yaitu HV, CC, LOC, perCM.

Setelah dilakukan pengujian dan didapatkan data hasil uji, maka selanjutnya akan dilakukan perbandingan nilai *maintainability index* antara MVVM dan MVC pada kedua aplikasi, yang dimana hasil akhir dari pengujian tersebut akan dilakukan analisis dan penarikan kesimpulan.

d. hasil dan analisis

Berdasarkan hasil pengujian sistem aplikasi yang dikembangkan pada tahap pengujian sistem, data yang didapat selanjutnya dilakukan analisis atau pengamatan data hasil uji, lalu dari data tersebut di ambil kesimpulan apakah selama proses pengujian dan hasil dari pengujian dapat menyelesaikan seluruh rumusan masalah di atas atau belum.

IV. HASIL DAN PEMBAHASAN

Hasil dari penelitian yang dilakukan adalah perbandingan nilai maintainability indeks antara pola desain MVVM dan MVC yang diuji pada dua aplikasi mobile yaitu "MyTodo" dan "Sisfo"

A. MYTODO

sistem aplikasi ini “MyTodo” Setelah dilakukan proses implementasi sistem aplikasi, dilanjutkan dengan pengujian maintainability index yang didukung dengan alat bantu pengujian “SonarQube” didapat data HV, CC, LOC, PerCM sebagai berikut:

Tabel 3 MVVM "MyTodo"

No	MyTodo (MVVM)	HV	CC	LOC	perCM (%)	MI
1	add_todo_page_mvvm.dart	1722	2	39	1.79	116
2	constant.dart	2845	3	63	1.96	103
3	dashboard_page_mvvm.dart	5325	2	105	0.64	98
4	database_helper.dart	2545	9	53	2.44	97
5	detail_todo_page_mvvm.dart	2998	3	65	1.04	111
6	main_mvvm.dart	564	2	17	3.85	97
7	text_field.dart	1582	0	36	1.96	116
8	todo_card.dart	1118	4	29	2.27	115
9	todo_model.dart	845	0	32	2.17	118
10	todo_viewmodel.dart	871	5	18	3.33	103
11	**System Level**	**2042**	**3**	**46**	**1.74**	**113**

Tabel 4 MVC "MyTodo"

No	MyTodo (MVC)	HV	CC	LOC	perCM (%)	MI
1	add_todo_page_mvvm.dart	1466	2	35	1.92	117
2	constant.dart	2845	3	63	1.96	103
3	dashboard_page_mvvm.dart	5658	0	115	1.16	99
4	database_helper.dart	2545	9	53	1.22	113
5	detail_todo_page_mvvm.dart	2935	3	63	2.13	100
6	main_mvvm.dart	375	2	13	0.00	98
7	text_field.dart	1596	0	36	1.92	116
8	todo_card.dart	1416	4	35	1.92	117
9	todo_model.dart	845	0	32	2.17	118
10	todo_viewmodel.dart	589	4	11	0.00	98
11	**System Level**	**2027**	**3**	**46**	**1.58**	**115**

Tabel 5 Maintainability "MyTodo"

No	Todo	MVVM	MVC
	Package Level		
1	add_todo_page_mvvm.dart	116	117
2	constant.dart	103	103
3	dashboard_page_mvvm.dart	98	99
4	database_helper.dart	97	113
5	detail_todo_page_mvvm.dart	111	100

6	main_mvvm.dart	97	98
7	text_field.dart	116	116
8	todo_card.dart	115	117
9	todo_model.dart	118	118
10	todo_modelview.dart / todo_controller.dart	103	98
	System Level	**113**	**115**

Pada tabel 3 dan tabel 4 secara berurutan menunjukkan hasil data HV, CC, LOC, dan perCm sebagai komponen Maintainability indeks yang didapat pada pengujian dengan "SonarQube". Lalu pada tabel tersebut juga menunjukkan nilai Maintainability indeks pada tingkat package dan sistem pada setiap pola desain.

Pada tabel 5 menunjukkan hasil pengukuran maintainability indeks dari sistem aplikasi "MyTodo" pada tingkat paket dan tingkat sistem. Hasil dari analisis tersebut menunjukkan pada tingkat paket dari total 10 paket yang dianalisis pada penelitian ini terdapat 4 paket yang menunjukkan bahwa MVC memiliki nilai yang lebih baik daripada MVVM, yaitu paket 1, 4, 6, dan 8, lalu ada 3 paket yang menunjukkan nilai yang sama yaitu paket 2, 7, dan 9. Pada tingkat sistem hasil pengujian menunjukkan MVC memiliki nilai yang lebih baik dari MVVM namun dengan selisih yang tidak signifikan.

B. SISFO

Sama seperti sistem aplikasi "MyTodo", sistem aplikasi ini "Sisfo" Setelah dilakukan proses implementasi sistem aplikasi dilanjutkan dengan pengujian maintainability index yang didukung dengan alat bantu pengujian "SonarQube" didapat data HV, CC, LOC, PerCM sebagai berikut:

Tabel 6 MVVM "Sisfo"

No	SISFO (MVVM)	HV	CC	LOC	perCM (%)	MI
1	add_izin.dart	6072	0	124	1.71	92
2	all_presence.dart	1666	0	29	3.64	87
3	bottom_navbar.dart	780	0	26	2.70	112
4	button_login.dart	1025	0	25	2.86	108
5	button_menu.dart	1348	0	39	1.85	117
6	constant.dart	2871	3	64	2.91	85
7	detail_presence.dart	576	0	16	4.00	95
8	header_fitur.dart	3202	0	68	1.96	102
9	home_page.dart	7762	0	162	0.79	91
10	izin.dart	5656	0	115	1.08	99
11	lorong.dart	785	2	26	2.78	110
12	lorongMV.dart	1342	1	21	3.33	99
13	main_app.dart	1493	0	32	1.89	119
14	main.dart	1943	1	30	2.08	116
15	monitoring.dart	6785	0	107	1.83	93
16	Onboarding1.dart	624	0	14	3.85	100
17	Onboarding2.dart	792	0	16	3.57	102
18	pop_in_up.dart	2174	0	62	2.20	102
19	private.dart	67	0	1	50.00	99
20	profile_user.dart	3665	0	73	1.90	101

21	report_user.dart	7442	0	139	1.41	93
22	santri.dart	1370	2	43	1.96	113
23	santriVM.dart	1672	1	31	2.22	114
24	sign_in.dart	2935	0	70	1.89	103
25	sign_up.dart	5148	0	75	1.63	103
26	text_field.dart	756	0	20	3.23	106
27	user.dart	1602	10	38	2.22	108
28	userVM.dart	2435	3	43	3.28	85
29	weekly_table.dart	2926	0	64	2.00	103
30	**System Level**	**2995**	**2**	**60**	**1.93**	**105**

Tabel 7 MVC "Sisfo"

No	SISFO (MVC)	HV	CC	LOC	perCM (%)	MI
1	add_izin.dart	6115	0	124	1.70	93
2	all_presence.dart	1560	0	29	3.70	86
3	bottom_navbar.dart	794	0	26	2.70	112
4	button_login.dart	1040	0	25	2.78	109
5	button_menu.dart	1348	0	39	1.85	117
6	constant.dart	2871	3	64	2.91	85
7	detail_presence.dart	576	0	16	4.17	92
8	header_fitur.dart	3162	0	68	1.98	102
9	home_page.dart	8709	0	172	0.76	89
10	izin.dart	5527	0	115	1.23	99
11	lorong.dart	3607	2	26	2.78	102
12	lorongMV.dart	1337	1	21	3.33	99
13	main_app.dart	1429	0	31	1.92	119
14	main.dart	1286	1	23	2.63	112
15	monitoring.dart	6752	0	107	1.83	93
16	Onboarding1.dart	624	0	14	3.85	100
17	Onboarding2.dart	785	0	16	3.45	105
18	pop_in_up.dart	2173	0	62	2.17	102
19	private.dart	67	0	1	50.00	99
20	profile_user.dart	4399	0	77	1.83	100
21	report_user.dart	7433	0	139	1.41	93
22	santri.dart	1370	2	43	1.96	113
23	santriVM.dart	1142	1	20	3.45	99
24	sign_in.dart	2382	0	63	2.08	103

25	sign_up.dart	5148	0	65	1.63	105
26	text_field.dart	771	0	20	3.13	108
27	user.dart	1602	10	38	2.22	108
28	userVM.dart	2059	3	38	1.72	116
29	weekly_table.dart	2926	0	64	2.00	103
30	**System Level**	**2923**	**2**	**57**	**1.87**	**106**

Tabel 8 Maintainability "Sisfo"

No	Sisfo	MVVM	MVC
	Package Level		
1	add_izin.dart	92	93
2	all_presence.dart	87	86
3	bottom_navbar.dart	112	112
4	button_login.dart	108	109
5	button_menu.dart	117	117
6	constant.dart	85	85
7	detail_presence.dart	95	92
8	header_fitur.dart	102	102
9	home_page.dart	91	89
10	izin.dart	99	99
11	lorong.dart	110	102
12	lorongMV.dart lorongController.dart	99	99
13	main_app.dart	119	119
14	main.dart	116	112
15	monitoring.dart	93	93
16	Onboarding1.dart	100	100
17	Onboarding2.dart	102	105
18	pop_in_up.dart	102	102
19	private.dart	99	99
20	profile_user.dart	101	100
21	report_user.dart	93	93
22	santri.dart	113	113
23	santriMV.dart santriController.dart	114	99
24	sign_in.dart	103	103
25	sign_up.dart	103	105
26	text_field.dart	106	108
27	user.dart	108	108

28	userMV.dart userController.dart	85	116
29	weekly_table.dart	103	103
	System Level	**105**	**106**

Pada tabel 6 dan tabel 7 secara berurutan menunjukkan hasil data HV, CC, LOC, dan perCm sebagai komponen Maintainability indeks yang didapat pada pengujian dengan “SonarQube”. Lalu pada tabel tersebut juga menunjukkan nilai Maintainability indeks pada tingkat package dan sistem pada setiap pola desain.

Pada tabel 8, menunjukkan hasil pengukuran maintainability indeks dari sistem aplikasi “Sisfo” pada tingkat paket dan tingkat sistem. Hasil dari analisis tersebut menunjukkan pada tingkat paket dari total 29 paket yang dianalisis pada penelitian ini terdapat 5 paket yang menunjukkan bahwa MVC memiliki nilai yang lebih baik daripada MVVM, yaitu paket 1, 4, 17, 26, dan 28, lalu ada 16 paket yang menunjukkan nilai yang sama yaitu paket 3, 5, 6, 8, 10, 12, 13, 15, 16, 18, 19, 21, 22, 24, 27, dan 29. Pada tingkat sistem hasil pengujian menunjukkan MVC memiliki nilai yang lebih baik dari MVVM namun dengan selisih yang juga tidak signifikan.

C. PEMBAHASAN

Analisis terhadap Halstead volume (HV) menunjukkan bahwa terdapat perbedaan dalam kompleksitas logika antara arsitektur MVVM dan MVC pada kedua aplikasi. Pada aplikasi "MyTodo", beberapa file seperti `dashboard_page_mvvm.dart` memiliki HV yang lebih tinggi pada arsitektur MVC (5658) dibandingkan dengan MVVM (5325), menunjukkan bahwa implementasi logika pada MVC lebih kompleks. Sebaliknya, file seperti `todo_viewmodel.dart` memiliki HV yang lebih tinggi pada MVVM (871) dibandingkan MVC (589), yang menandakan bahwa logika MVVM di file ini lebih kompleks. Pada aplikasi "Sisfo", perbedaan HV juga terlihat signifikan pada beberapa file, seperti `home_page.dart` yang memiliki HV sangat tinggi pada MVC (8709) dibandingkan dengan MVVM (7762). Hal ini menunjukkan bahwa arsitektur yang digunakan mempengaruhi kompleksitas logika yang diimplementasikan di dalam aplikasi.

Cyclomatic Complexity (CC) mengukur kompleksitas alur kontrol dalam suatu program, yang ditentukan oleh jumlah jalur eksekusi yang mungkin terjadi. Hasil pengujian menunjukkan bahwa sebagian besar file dalam kedua aplikasi memiliki nilai CC yang sama antara MVVM dan MVC, menunjukkan bahwa kompleksitas alur kontrol tidak terlalu terpengaruh oleh arsitektur yang digunakan. Namun, pada aplikasi "MyTodo", terdapat perbedaan CC pada file `dashboard_page_mvvm.dart` di mana MVVM memiliki nilai CC 2, sementara MVC memiliki nilai 0. Ini menunjukkan bahwa versi MVC lebih sederhana atau tidak memiliki kondisi pengambilan keputusan yang rumit. Di aplikasi "Sisfo", mayoritas file memiliki nilai CC yang sama di kedua arsitektur, kecuali beberapa file seperti `user.dart` yang memiliki CC lebih tinggi (10), menunjukkan adanya logika percabangan yang lebih kompleks.

Dari hasil pengujian, terlihat bahwa sebagian besar file dalam kedua aplikasi memiliki jumlah baris kode yang sama antara arsitektur MVVM dan MVC, menunjukkan bahwa struktur dasar kode tetap konsisten terlepas dari arsitektur yang digunakan. Namun, terdapat beberapa perbedaan yang menonjol, seperti pada file `dashboard_page_mvvm.dart` dalam aplikasi "MyTodo", di mana MVC memiliki lebih banyak baris kode (115) dibandingkan MVVM (105). Hal ini mungkin menunjukkan adanya tambahan fitur atau logika dalam versi MVC. Pada aplikasi "Sisfo", perbedaan LoC juga terlihat pada beberapa file seperti `main.dart`, di mana MVVM memiliki 30 baris kode sedangkan MVC memiliki 23, menunjukkan perbedaan dalam implementasi antara kedua arsitektur.

Percent line of comment (PerCM) mengukur persentase baris kode yang berisi komentar, yang memberikan indikasi tentang seberapa baik kode tersebut didokumentasikan. Hasil pengujian menunjukkan bahwa pada aplikasi "MyTodo", file `main_mvvm.dart` memiliki nilai perCM yang lebih tinggi pada MVVM (3.85) dibandingkan dengan MVC (0.00), menunjukkan bahwa kode pada versi MVVM lebih baik didokumentasikan. Sebaliknya, pada aplikasi "Sisfo", sebagian besar file memiliki nilai perCM yang mirip antara MVVM dan MVC. Namun, ada beberapa perbedaan, seperti pada file `userController.dart`, di mana MVVM memiliki nilai perCM 3.28 dibandingkan dengan MVC 1.72. Ini menunjukkan bahwa dokumentasi pada arsitektur MVVM lebih lengkap dibandingkan dengan MVC untuk file ini.

Hasil pengujian maintainability index pada aplikasi "MyTodo" menunjukkan bahwa pada tingkat paket, dari 10 paket yang dianalisis, MVC memiliki nilai lebih baik pada 4 paket, MVVM lebih baik pada 3 paket, dan 3 paket lainnya memiliki nilai yang sama. Pada tingkat sistem, MVC sedikit lebih unggul dengan nilai 115 dibandingkan MVVM yang bernilai 113, meskipun perbedaannya tidak signifikan. Hal ini menunjukkan bahwa kedua arsitektur ini hampir setara dalam hal maintainability untuk aplikasi "MyTodo".

Pada aplikasi "Sisfo", hasil pengukuran maintainability index pada tingkat paket dari 29 paket menunjukkan bahwa MVC lebih baik pada 5 paket, MVVM lebih baik pada 8 paket, dan 16 paket memiliki nilai yang sama. Di tingkat sistem, MVC juga sedikit lebih unggul dengan nilai 106 dibandingkan MVVM yang bernilai 105, namun perbedaannya juga tidak signifikan. Ini menunjukkan bahwa maintainability kedua arsitektur ini hampir setara untuk aplikasi "Sisfo".

Secara keseluruhan pada penelitian ini menunjukkan bahwa terdapat perbedaan signifikan antara MVVM dan MVC dalam hal komponen maintainability indeks, yaitu pada Halted Volume (HV), Cyclomatic Complexity (CC), Line of Code (LoC), dan Percent line of comment (PerCM). Pada Halstead volume menunjukkan tingkat kompleksitas logika dalam sebuah program, pada sistem “MyTodo” hasil pengujian menunjukkan bahwa pada beberapa paket MVVM memiliki nilai HV yang lebih tinggi dibandingkan MVC yang mengindikasi bahwa MVVM

memiliki logika pemrograman yang kompleks. Namun, ada juga beberapa file yang menunjukkan bahwa MVC memiliki nilai yang lebih tinggi dibandingkan MVVM, ini menunjukkan bahwa kompleksitas logika MVVM bisa lebih tinggi tergantung pada penggunaan ViewModel. Hal ini menunjukkan bahwa MVVM cenderung lebih baik untuk file dengan peran spesifik, sementara MVC cenderung lebih baik untuk struktur tertentu yang lebih sederhana.

Pada hasil pengujian Cyclomatic Complexity (CC) menunjukkan bahwa sebagian besar file dalam kedua sistem memiliki nilai CC yang hampir sama antara MVVM dan MVC. Namun, ada beberapa juga file yang menunjukkan MVVM lebih unggul dibanding MVC, ini menunjukkan bahwa MVC memiliki alur kontrol yang lebih sederhana. Maka dari itu, MVC dapat digunakan ketika sebuah aplikasi membutuhkan alur kontrol yang sederhana tanpa banyak percabangan program. Disisi lain, MVVM memberikan struktur yang kompleks dan lebih fleksibel untuk aplikasi dengan kebutuhan logika yang lebih kompleks.

Dalam hal Line of Code (LoC), penelitian ini menunjukkan bahwa sebagian besar file memiliki jumlah baris kode yang sama antara MVVM dan MVC, tetapi ada beberapa perbedaan penting. Misalnya, pada "MyTodo", MVC memiliki lebih banyak baris kode dibandingkan MVVM, ini menunjukan adanya tambahan fitur atau logika pada system aplikasi tersebu. MVVM umumnya mengurangi jumlah baris kode pada View dengan memasukkan logika ke ViewModel, membuat kode lebih bersih dan mudah dikelola. Selain itu, pada Percent line of comment (PerCM) menunjukkan bahwa MVVM memiliki dokumentasi yang lebih baik di beberapa file, seperti main_mvvm.dart pada aplikasi "MyTodo", di mana nilai PerCM untuk MVVM lebih tinggi (3.85) dibandingkan MVC (0.00).

Secara keseluruhan berdasarkan analisis hasil pengujian maintainability index pada aplikasi "MyTodo" dan "Sisfo", dapat disimpulkan bahwa baik arsitektur Model-View-ViewModel (MVVM) maupun Model-View-Controller (MVC) menunjukkan maintainability yang hampir setara pada tingkat sistem, meskipun terdapat variasi pada tingkat paket. Pada aplikasi "MyTodo", MVC sedikit lebih unggul dengan nilai maintainability index 115 dibandingkan MVVM yang bernilai 113. Sementara itu, pada aplikasi "Sisfo", MVC juga menunjukkan nilai sedikit lebih tinggi dengan nilai 106 dibandingkan MVVM yang bernilai 105.

Berdasarkan hasil pengujian tersebut menunjukkan hasil akhir pengujian yang sama dengan penelitian sebelumnya[3], pada penelitian tersebut melakukan pengujian antara pola desain MVP dan anti Pattern atau tanpa pola desain. Pada penelitian tersebut menunjukkan hasil akhir pengujian MVP memiliki nilai 112 dan anti Pattern 109, ini menunjukkan bahwa untuk mendapatkan maintainability yang baik dalam pengembangan sistem aplikasi mobile, tidak selalu terfokus pada pemilihan pola desain yang digunakan.

V. KESIMPULAN

Kesimpulan dari penelitian ini adalah bahwa MVVM atau MVC dapat dijadikan acuan bagi pengembang dalam memilih pola desain arsitektur untuk aplikasi perangkat lunak berbasis mobile, karena keduanya memiliki kemampuan maintainability yang hampir sama. Namun data hasil penelitian pada setiap komponen maintainability indeks menunjukkan bahwa MVVM lebih baik digunakan dalam memisahkan logika bisnis dari logika interface dan mendukung dokumentasi yang lebih baik. Sebaliknya, MVC memiliki struktur yang sederhana dan mudah dipahami, pola desain ini cocok untuk aplikasi dengan logika yang sederhana. Hal tersebut didasari dari data hasil pengujian pada komponen Halstead volume (HV), Cyclomatic Complexity (CC) yang menunjukkan MVVM memiliki nilai yang lebih tinggi dibanding MVC.

Selanjutnya, karena hasil akhir dari penelitian ini menunjukkan kesamaan dengan hasil akhir penelitian sebelumnya[3], maka dapat di simpulkan bahwa untuk mendapatkan tingkat maintainability yang baik dalam pengembangan sistem aplikasi mobile, tidak selalu terfokus pada pemilihan pola desain yang digunakan. Oleh karena itu berdasarkan hasil penelitian ini, memberikan beberapa saran penelitian lebih lanjut yang dapat dilakukan untuk memperdalam pemahaman tentang bagaimana mendapatkan nilai maintainability yang baik dalam pengembangan sebuah sistem aplikasi. Penelitian lebih lanjut dapat dilakukan eksplorasi pola desain lainnya seperti MVP (Model-View-Present), atau Clean Architectur yang diharapkan dapat memberikan wawasan lebih luas mengenai pengaruh pola desain terhadap tingkat maintainability sebuah sistem aplikasi terutama pada perangkat mobile, dapat juga dilakukan penelitian lebih lanjut mengenai perbandingan komponen-komponen penunjang Maintainability Indeks (MI).

REFERENSI

[1] Simon kemp, “DIGITAL 2024: INDONESIA,” Datareportal.com. Accessed: Jun. 13, 2024. [Online]. Available: https://datareportal.com/reports/digital-2024-indonesia

[2] Ginanjar Prabowo, Hatma Suryotrisongko, and Aris Tjahyanto, *The 2nd 2018 International Conference on Electrical Engineering and Informatics (ICELTICs 2018) : “A Tale of Two Development Approach: Empirical Study on The Maintainability and Modularity of Android Mobile Application with Anti-Pattern and Model-View-Presenter Design Pattern” : proceedings : Banda Aceh, Aceh, Indonesia, September 19-20, 2018*.

[3] G. Bagus, V. Putraadinatha, D. Dwi, J. Suwawi, and S. Y. Puspitasari, “Pengaruh Design Pattern Terhadap Maintainability Aplikasi Mobile,” vol. 8, no. 5, p. 10954, 2021.

[4] N. Rachael, “MOBILE DESIGN PATTERNS: A REVIEW OF MVC, MVP, MVVM, MOBILE DESIGN PATTERNS: A REVIEW OF MVC, MVP, MVVM, AND VIPER AND VIPER.”

[5] A. H. Zakaria, I. Kadek, and D. Nuryana, “MVVM Perangkat Lunak Android dan Analisis Arsitektur MVP dengan Studi Kasus Aplikasi iTourism”, [Online]. Available: https://api.imgflip.com/

[6] PT Revolusi Cita Edukasi, “MVVM,” revou.co.

[7] Vianka Tetiana, Dana Sulistiyo Kusumo, and Monterico Andrian, “Analisis Pengaruh Pola Arsitektur Model View View Model (MVVM) terhadap Kinerja Aplikasi Mobile dengan Menerapkan Application Programming Interface

(API) Covid," *e-Proceeding of Engineering : Vol.10, No.3 Juni 2023*, vol. 10, 2023.
[8] Martin Fowler, "Presentation Model."
[9] Mazewaxie and Pngdeity, "Model–view–controller," Wikipedia encyclopedia.
[10] R. Ganang Atmaja, B. Priyambadha, and F. Pradana, "Pembangunan Kakas Bantu Untuk Mengukur Maintainability Index Pada Perangkat Lunak Berdasarkan Nilai Halstead Metrics dan McCabe's Cyclomatic Complexity," 2019. [Online]. Available: http://j-ptiik.ub.ac.id
[11] M. Deta Aditya, M. Susanty, and I. Artikel, "Studi Komparasi Maintainability Antara Aplikasi yang Dikembangkan dengan Framework Flutter dan React Native," *JURNAL INFORMATIKA*, vol. 9, no. 2, 2022, [Online]. Available: http://ejournal.bsi.ac.id/ejurnal/index.php/ji