JITET (Jurnal Informatika dan Teknik Elektro Terapan)
Vol. 14 No. 2, pISSN: 2303-0577 eISSN: 2830-7062 http://dx.doi.org/10.23960/jitet.v14i2.9161

IMPLEMENTASI ARSITEKTUR MODEL-VIEW-VIEWMODEL (MVVM) DALAM PENGEMBANGAN SISTEM DIL-MICLEARN BERBASIS MOBILE

Putu Suardana[1], Ni Wayan Marti[2*] , Ketut Agus Seputra[3]

[1,2,3] Jurusan Teknik Informatika, Universitas Pendidikan Ganesha; Jl. Udayana No. 11 Singaraja-Bali

Keywords:
DIL-MicLearn; Model-View-ViewModel; e-learning; Aplikasi Mobile.

Corespondent Email:
wayan.marti@undiksha.ac.id

Abstrak. Penelitian ini bertujuan merancang sistem e-learning DIL-MicLearn berbasis mobile dengan mengimplementasikan arsitektur Model-View-ViewModel (MVVM). Pengembangan sistem dilakukan dengan pendekatan Software Development Life Cycle (SDLC) menggunakan metode waterfall, melalui Android Studio, bahasa pemrograman Kotlin, Jetpack Compose sebagai framework, dan MySQL sebagai sistem manajemen basis data. Hasil penelitian ini berupa sistem DIL-MicLearn berbasis mobile yang lebih terstruktur dan efisien dengan memisahkan logika bisnis dari antarmuka pengguna. Implementasi fitur notifikasi memungkinkan mahasiswa menerima pemberitahuan secara langsung melalui perangkat smartphone, mempermudah mereka mengingat jadwal tes sumatif dan informasi penting lainnya tanpa harus masuk ke sistem secara manual. Selain itu, penggunaan WorkManager dalam penjadwalan notifikasi memastikan notifikasi dapat dijalankan tepat waktu sesuai jadwal yang telah ditentukan, sehingga meningkatkan pengalaman pengguna dan mendukung tujuan pembelajaran yang efektif.

Abstract. *This study aims to design DIL-MicLearn, a mobile-based e-learning system, by implementing the Model-View-ViewModel (MVVM) architecture. The system development followed the Software Development Life Cycle (SDLC) approach utilizing the Waterfall methodology. The development environment included Android Studio, the Kotlin programming language, Jetpack Compose as the user interface framework, and MySQL as the Database Management System (DBMS). The outcome of this research is a mobile-based DIL-MicLearn system that exhibits enhanced structure and efficiency by decoupling business logic from the user interface. The implementation of a notification feature enables students to receive real-time alerts on their mobile devices, facilitating the tracking of summative test schedules and other critical information without requiring manual system logins. Furthermore, the utilization of WorkManager for notification scheduling ensures timely execution according to predetermined parameters, thereby enhancing the user experience and supporting effective learning objectives.*

1. PENDAHULUAN

Era revolusi industri 4.0, ditandai dengan perkembangan pesat teknologi informasi, telah mempengaruhi berbagai aktivitas manusia, termasuk pendidikan. Salah satu inovasi penting adalah e-learning, yang memanfaatkan teknologi komunikasi berbasis elektronik untuk pembelajaran jarak jauh. E-learning memudahkan dosen dan mahasiswa dalam proses pembelajaran, namun penggunaannya

masih belum optimal, terutama karena fitur yang digunakan terbatas pada pengunduhan materi atau tugas [1], [2].

Saat ini telah dikembangkan *Dynamic Intellectual Learning-Micro Learning* (*DIL-MicLearn*) berbasis web. *DIL-MicLearn* adalah sistem *e-learning* yang mengintegrasikan teknik *microlearning* [3], [4] untuk mencapai ketuntasan belajar [5]. Sistem ini memungkinkan mahasiswa mempelajari materi secara bertahap hingga tuntas dan melanjutkan ke konsep berikutnya [3]. Sistem *DIL-MicLearn* dikembangkan dengan *framework* Laravel 9 dengan skrip PHP 8 sebagai *backend*, Vue.js versi 3.3.0 sebagai *frontend*, dan menggunakan MySQL 10.4.25 sebagai model pengelolaan basis data (DBMS) [6]. Sistem DIL-MicLearn mengimplementasikan REST API untuk memungkinkan sistem berkomunikasi dengan server untuk mengambil atau mengirimkan data. Hal tersebut memungkinkan sistem ini dapat dikembangkan dari website ke berbagai platform [7], [8].

Sistem *DIL-MicLearn* yang berbasis website ini memang telah berjalan sesuai dengan konsep dari ketuntasan belajar [5]. Namun pengguna khususnya mahasiswa harus terus memantau website *DIL-MicLearn* untuk mendapatkan informasi terbaru, salah satu contohnya pengerjaan tes sumatif yang telah dijadwalkan pengerjaanya oleh dosen pengampu [3]. Tidak adanya pemberitahuan dari website terkait jadwal terbukanya pengerjaan tes sumatif tersebut menimbulkan kemungkinan mahasiswa dapat melewatkan tes tersebut. Dengan adanya pemberitahuan langsung pada perangkat, membantu pengguna khususnya mahasiswa melakukan proses pembelajaran [9]. Selain itu, beberapa aktivitas yang penting seperti masuk ke dalam kelas secara mandiri oleh mahasiswa, melihat daftar semua kelas yang belum diikuti dan terdaftar dalam sistem, dan melihat informasi detail dari kelas yang ingin diikuti mahasiswa belum terimplementasi dalam sistem *DIL-MicLearn* berbasis website [5]. Jika mahasiswa ingin masuk ke dalam kelas yang diinginkannya, mahasiswa harus menghubungi dosen pengampu dan dosen pengampu harus menghubungi admin terlebih dahulu untuk memasukkan mahasiswa tersebut ke dalam kelasnya. Sehingga, permasalahan tersebut harus ditangani dengan mengimplementasikan fitur tersebut untuk mempermudah pengguna baik mahasiswa, dosen, ataupun admin dalam melaksanakan aktivitasnya pada sistem ini.

Dalam proses integrasi website ke versi android, yang diinginkan *client* adalah aplikasi berkualitas dengan sistem yang kompleks [10] namun waktu pengembangan yang singkat [11], [12]. Dengan permintaan tersebut, diperlukannya suatu pola arsitektur perancangan yang tepat untuk merancang sebuah aplikasi berbasis android. Pola arsitektur aplikasi berbasis android/mobile adalah ‘organisasi’ kode secara menyeluruh [10]. Organisasi disini dapat diartikan dengan pemisahan antara alur bisnis dengan tampilan antarmuka dari sebuah aplikasi. MVVM merupakan pola arsitektur yang sempurna untuk menangani pengembangan sistem dengan alur bisnis yang kompleks [11] untuk aplikasi berbasis android.

Berdasarkan masalah yang telah dipaparkan, untuk itu perlu dikembangkan aplikasi *DIL-MicLearn* dalam versi *mobile* dengan mengimplementasikan arsitektur MVVM. Penelitian ini dilakukan untuk mengoptimalkan kemampuan sistem *DIL-MicLearn* sehingga dapat dimanfaatkan secara efektif dalam pembelajaran dengan penerapan metode belajar tuntas. Dengan demikian, hasil penelitian ini diharapkan meningkatkan kualitas proses pembelajaran berbasis teknologi di Undiksha.

2. TINJAUAN PUSTAKA

2.1 Sistem *DIL-MicLearn*

DIL-MicLearn merupakan e-learning berbasis website yang mengkolaborasikan sistem *dynamic intellectual learning* dan teknik *microlearning* [3], [4] untuk pencapaian ketuntasan belajar mahasiswa. Sistem ini telah diimplementasikan pada pembelajaran online mata kuliah Basis Data di Prodi Ilmu Komputer, Universitas Pendidikan Ganesha [6]. Dengan penerapan konsep ketuntasan belajar dalam sistem ini membuat mahasiswa dapat mengeksplorasi materi pembelajaran sesuai dengan kemampuannya [3], [5]. Mahasiswa dapat mengikuti pembelajaraan secara lancar tanpa harus terhambat oleh mahasiswa lain yang masih memiliki masalah dalam materi pembelajaran. Dengan demikian, dapat meningkatan kemandirian dari suatu mahasiswa [13].

2.2 Model-View-ViewModel (MVVM)

MVVM merupakan sebuah pola arsitektur pengembangan aplikasi yang lebih modern dari Model-View-Controller (MVC) dengan tujuan inti memisahkan kode tampilan antarmuka dengan kode logika bisnis dari sebuah aplikasi [14]. Hal yang dimaksud adalah baris kode logika bisnis yang ada dalam suatu View dipindahkan ke dalam ViewModel yang terpisah. Pola arsitektur MVVM memiliki beberapa komponen utama yaitu Model, View, ViewModel. Masing-masing dari komponen tersebut memiliki peran dan tanggung jawab [15]. Model bertanggung jawab atas penyediaan data yang diperlukan sekaligus merepresentasikan data tersebut dalam logika bisnis. Model ini bisa berisi Plain Old Java Object (POJO) atau Kotlin Data Class. View sendiri memiliki tanggung jawab untuk semua hal yang berhubungan dengan tampilan antarmuka. Pola arsitektur MVVM dianggap sebagai pola arsitektur yang efektif dalam pengembangan aplikasi berbasis mobile karena memiliki beberapa sifat yang menjadi keunggulan yaitu bersifat *low coupling* dan *high cohesion* [10]–[12]. Sifat *low coupling* yang berarti bagian dari komponen memiliki ketergantungan yang rendah dengan komponen lain yang berbeda. Sedangkan sifat *high cohesion* artinya setiap komponen memiliki tujuan yang jelas dan berpegang teguh dengan tujuan tersebut, sehingga kode dapat lebih mudah dikerjakan dan digunakan kembali.

3. METODE PENELITIAN

Penelitian ini menggunakan pendekatan *Software Development Life Cycle* (SDLC) yang merupakan pendekatan terstruktur dalam pengembangan perangkat lunak untuk membuat atau mengubah sistem aplikasi sesuai kebutuhan [16], [17]. Bagian ini menjelaskan bagaimana penelitian dilakukan meliputi data penelitian, peralatan penelitian, dan tahapan penelitian.

A. Data Penelitian

Data yang digunakan dalam penelitian ini diperoleh melalui wawancara dengan narasumber, yaitu dosen pengampu mata kuliah dan pengembang sistem DIL-MicLearn sebelumnya. Data tersebut dikumpulkan untuk mengidentifikasi kebutuhan sistem yang akan dikembangkan dan menentukan pendekatan implementasi, termasuk arsitektur pengembangan sistem *mobile* yang paling sesuai. Tahap selanjutnya adalah menganalisis data untuk merancang sistem DIL-MicLearn berbasis *mobile* yang mendukung kebutuhan fungsional dan non-fungsional pengguna.

B. Peralatan Penelitian

Sarana yang digunakan dalam pelaksanaan penelitian ini yaitu:

1) Perangkat keras berupa laptop dengan spesifikasi:

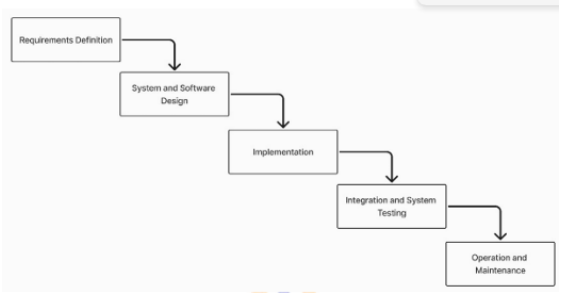
- Prosesor AMD Athlon Gold 3150U
- Memori RAM 12GB
- Penyimpanan SSD 512GB
- Kartu Grafis AMD Radeon Graphics

2) Perangkat lunak yang digunakan:

- Windows 11
- Android Studio
- Android Virtual Device
- Figma
- Postman

C. Tahapan Penelitian

Penelitian ini menggunakan metode *Waterfall*, yang dipilih karena pengerjaannya sistematis, terstruktur, dan sesuai untuk sistem DIL-MicLearn yang memiliki kebutuhan konstan dengan spesifikasi jelas sejak awal. Menurut Rosa A.S dan Shalahuddin dalam bukunya yang berjudul "Rekayasa Perangkat Lunak Terstruktur Dan Berorientasi Objek" metode ini terbagi menjadi beberapa tahapan yang dilihat pada Gambar 1 [18].



Gambar 1. Alur pengembangan perangkat lunak menggunakan metode *Waterfall*

1. Requirements Definition: Pada tahap ini, peneliti memulai dengan wawancara dan diskusi bersama beberapa narasumber yang terlibat langsung dalam sistem DIL-MicLearn

untuk mendefinisikan kebutuhan sistem. Narasumber tersebut meliputi dosen pengampu mata kuliah, yang merupakan pengguna utama sistem, dengan tujuan menggali kebutuhan, spesifikasi, dan batasan sistem berbasis *mobile*, serta pengembang sebelumnya, untuk memperoleh informasi teknis seperti dokumentasi API, database, dan aspek teknis lainnya. Hasil dari wawancara ini kemudian digunakan dalam penyusunan dokumen *Software Requirements Specification* (SRS) DIL-MicLearn sebagai panduan dalam pengembangan sistem.

2. System and Software Design: Setelah memperoleh informasi mengenai kebutuhan sistem, peneliti melanjutkan dengan merancang desain antarmuka pengguna untuk sistem DIL-MicLearn, yang menggambarkan tampilan sistem pada perangkat pengguna. Perancangan ini melibatkan dua tahapan utama, yaitu *Low-Fidelity Design* dan *High-Fidelity Design*, yang dikerjakan menggunakan perangkat lunak Figma sebagai alat bantu.

3. Implementation: Pada tahapan ini, seluruh rancangan yang telah dibuat, baik berupa dokumen maupun desain, diimplementasikan ke dalam kode pemrograman dan diuji coba pada sistem yang telah dikembangkan. Beberapa kegiatan utama yang dilakukan meliputi pengambilan data dari API menggunakan perangkat lunak Postman, implementasi desain antarmuka menggunakan kerangka kerja Jetpack Compose, integrasi API ke dalam kode untuk menampilkan data yang telah diuji sebelumnya melalui Postman, serta penerapan arsitektur MVVM dalam pengembangan sistem.

4. Integration and System Testing: Pada tahap ini, sistem yang telah selesai dikembangkan masuk ke dalam proses *system testing*, yang melibatkan dua kali uji coba. Pertama, dilakukan *User Interface Test* (UI Test) secara keseluruhan atau *black box testing* untuk memastikan bahwa logika antarmuka pengguna berfungsi sesuai dengan rancangan sistem DIL-MicLearn. Kedua, dilakukan pengujian terhadap pengguna dengan melibatkan kelompok kecil sebagai sampel, guna memastikan aplikasi memiliki nilai *usability* yang baik. Pada tahap ini, metode *System Usability Scale* (SUS) digunakan untuk mengukur nilai *usability* dari aplikasi yang telah dikembangkan.

5. Operation and Maintenance: Tahapan ini merupakan tahap akhir dalam penelitian ini, di mana sistem dengan *output* aplikasi berbasis *mobile*/Android (.apk) telah siap digunakan oleh banyak pengguna sesuai dengan target yang telah ditentukan. Aplikasi telah diselesaikan dengan kondisi minim, bahkan tanpa adanya *bug* atau *crash* yang mengganggu selama penggunaan aplikasi

4. HASIL DAN PEMBAHASAN

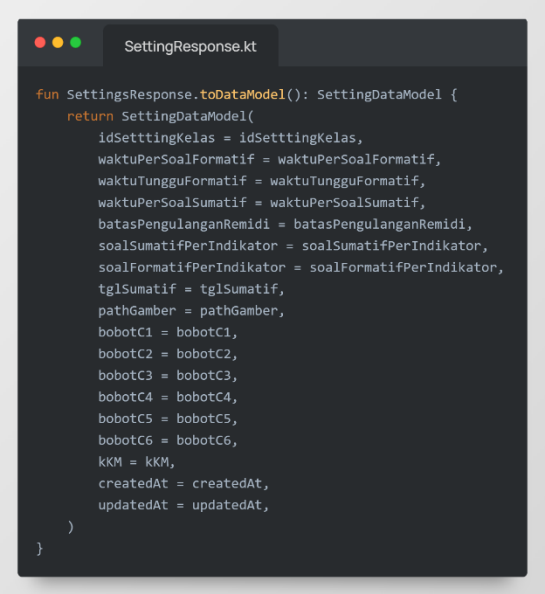
4.1 Penerapan Asitektur *MVVM*

Penerapan desain arsitektur MVVM memiliki tujuan memisahkan kode bagian antar muka yang ditangani oleh komponen *View* (*Activity/Fragment*) dan bagian logika bisnis yang ditangani oleh komponen *ViewModel*. Proses ini menggunakan bahasa pemrograman Kotlin dengan *framework* Jetpack Compose, karena dengan *framework* ini memberikan kemudahan untuk membuat sebuah antarmuka dengan *Compose UI System* [15].

Model merupakan entitas atau objek yang digunakan untuk merepresentasikan dan menyimpan informasi atau data dalam aplikasi. Meskipun model berperan dalam pengelolaan data, ia tidak bertanggung jawab atas manipulasi data tersebut [15]. Dalam pengembangan sistem DIL-MicLearn berbasis *mobile*, data yang diperoleh dari API DIL-MicLearn dalam format JSON diolah melalui proses *deserialize* menggunakan *Plain Old Java Object* (POJO) untuk menghasilkan kelas *response*. Kelas ini kemudian ditransformasi menggunakan fungsi *toDataModel* menjadi sebuah kelas data model yang berperan sebagai *Model* dalam konsep MVVM.

Sebagai contoh, pada Gambar 2 ditampilkan proses transformasi dari *SettingsResponse* menjadi *SettingsDataModel* melalui fungsi *toDataModel*. Fungsi ini bertugas mengonversi data dari format eksternal (*response* API) ke format internal yang sesuai dengan kebutuhan aplikasi. Setiap properti dari *SettingsResponse* dipetakan secara spesifik ke properti yang relevan dalam *SettingsDataModel* untuk memastikan integritas data yang digunakan dalam sistem.

Walaupun *SettingsResponse* berfungsi untuk menangkap dan merepresentasikan data dari API, kelas ini bukanlah model dalam konsep MVVM. *SettingsResponse* harus ditransformasikan menjadi *SettingsDataModel*, yang kemudian dapat digunakan untuk mengelola data di dalam aplikasi sesuai kebutuhan.



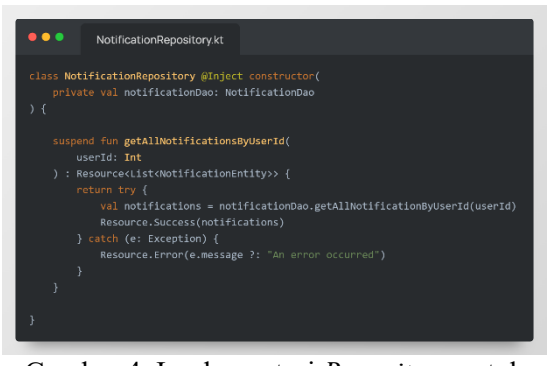
Gambar 2. Fungsi *toDataModel* untuk memetakan setiap properti dalam SettingsResponse ke dalam SettingsDataModel

Agar data dari API atau sumber data internal seperti *database* lokal dapat diakses dan diproses menjadi model yang siap digunakan, dibutuhkan komponen *repository*. *Repository* berfungsi sebagai penghubung antara sumber data, baik eksternal (seperti API) atau internal (seperti *database* lokal) dan *ViewModel*. Tugas *repository* adalah mengambil data dari sumber tersebut, memprosesnya, dan mengembalikannya dalam bentuk model yang siap digunakan oleh *ViewModel*. Implementasi penggunaan *repository* ditunjukan pada Gambar 3 dan Gambar 4, yang masing-masing menunjukan proses penyimpanan data ke *database* lokal dan pengambilan data dari *database* lokal.

Gambar 3. Implementasi *Repository* untuk menyimpan data pada *database* lokal

Pada Gambar 3, fungsi *createNotification*, repository bertanggung jawab untuk mengelola penyimpanan data notifikasi ke dalam *database* lokal. Fungsi ini menerima parameter yang diperlukan untuk membuat objek *NotificationEntity*, yang kemudian disimpan ke dalam *database* menggunakan *notificationDao* dengan metode *insertNotification*. *Repository* memastikan bahwa proses penyimpanan berjalan dengan baik. Jika penyimpanan berhasil, fungsi akan mengembalikan ID notifikasi yang baru disimpan dalam bentuk *Resource.Success*. Namun, jika terjadi kesalahan, *repository* akan menangani pengecualian dan mengembalikan pesan kesalahan melalui *Resource.Error*.

Selain berfungsi untuk menyimpan data, *repository* juga dapat mengambil data dari database lokal, memprosesnya, dan mengembalikannya dalam bentuk *model* yang siap digunakan oleh *ViewModel*.



Gambar 4. Implementasi *Repository* untuk mengambil data dalam database lokal

Pada Gambar 4, fungsi dalam repository digunakan untuk mengambil data notifikasi dari database lokal berdasarkan parameter *userId*. Fungsi ini memanfaatkan metode *getAllNotificationsByUserId* dari *notificationDao* untuk melakukan query pada database dan mengambil daftar notifikasi dalam bentuk *NotificationEntity*. Setelah data diperoleh, repository mengemasnya ke dalam objek *Resource.Succes* untuk menunjukkan keberhasilan proses pengambilan data. Sebaliknya, jika terjadi kesalahan selama proses pengambilan data, repository menangani pengecualian dan mengembalikan pesan kesalahan melalui objek *Resource.Error*.

Selanjutnya, pada *ViewModel*, data yang telah disiapkan oleh *repository* diproses lebih lanjut dengan logika bisnis agar siap ditampilkan pada *View*. Sebagai contoh, seperti yang ditunjukkan pada Gambar 5, fungsi *getNotificationByUserId* di *ViewModel* menggunakan *coroutine* di *viewModelScope* untuk menjalankan proses pengambilan data secara asinkron dari *repository*, sehingga operasi tersebut tidak mengganggu antarmuka pengguna. Data yang diperoleh dari *repository* kemudian diperbarui ke suatu variabel, yang berupa *MutableStateFlow*, untuk mengelola status data secara reaktif. Status awal data diatur ke *Resource.Loading()* sebagai penanda bahwa data sedang dalam proses pengambilan, dan hasil akhirnya, baik sukses maupun error, diperbarui ke *StateFlow* agar dapat langsung diamati oleh *View*. Dengan pendekatan ini, *ViewModel* memisahkan logika pengambilan data dari *View* sekaligus memastikan bahwa data yang diakses oleh *View* telah tersedia dalam format yang siap digunakan.

NotificationViewModel.kt

```
@HiltViewModel
class NotificationViewModel
@Inject constructor(
    private val repository: NotificationRepository,
) : ViewModel() {

    private val _notifications =
        MutableStateFlow<Resource<List<NotificationEntity>>>(
            Resource.Nothing()
        )
    val notifications :
            StateFlow<Resource<List<NotificationEntity>>> = _notifications

    fun getNotificationByUserId(userId: Int) {
        viewModelScope.launch(Dispatchers.IO) {
            _notifications.value = Resource.Loading()
            val result = repository.getAllNotificationsByUserId(userId)
            _notifications.value = result

            if (result is Resource.Error) {
                Log.e("Get Notifications", result.message.toString())
            }
        }
    }

}
```

Gambar 5. Implementasi *ViewModel* dengan logika bisnis mengambil data notifikasi berdasarkan ID

Pada bagian *View*, data dari *ViewModel* diambil menggunakan *StateFlow* dan ditampilkan ke antarmuka. Proses pengambilan data ini dikontrol oleh *LaunchedEffect*, yang memastikan *View* dapat mendeteksi apakah data dari *ViewModel* sudah tersedia atau belum. LaunchedEffect tidak melakukan sinkronisasi otomatis, melainkan memantau kondisi tertentu yang memicu pengambilan data dari *ViewModel*. Implementasi penampilan data dalam *View* ini dapat dilihat pada Gambar 6.

NotificationsList.kt

```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun NotificationsList(
    notificationViewModel: NotificationViewModel = hiltViewModel(),
) {

    /* Untuk mendapatkan list notifikasi */
    val notificationsState by notificationViewModel.notifications.collectAsState()
    var notifications by remember { mutableStateOf<List<NotificationEntity>>(listOf()) }
    var myNotif by remember { mutableStateOf<List<NotificationEntity>>(listOf()) }
    LaunchedEffect(notificationsState) {
        when (notificationsState) {
            is Resource.Success -> {
                notifications = notificationsState.data!!
                myNotif = notifications.filter { it.userId == userId }
            }

            is Resource.Error -> {
                val errorMessage = (notificationsState as Resource.Error).message
            }

            else -> {}
        }
    }

    /* Variable myNotif akan masuk pada kondiri perulangan untuk menampilkan daftar dari notifikasi */
}
```

Gambar 6. Implementasi dalam menampilkan data pada *View*

Ketika *LaunchedEffect* mendeteksi bahwa data sudah tersedia, *View* akan melakukan pengkondisian. Jika data yang diterima berstatus *Success*, materi dan daftar materi yang

berhasil diambil akan ditampilkan kepada pengguna. Sebaliknya, jika data berstatus *Error*, *View* akan menangani skenario kesalahan dengan menampilkan pesan kosong atau informasi *error* sesuai kondisi yang terjadi.

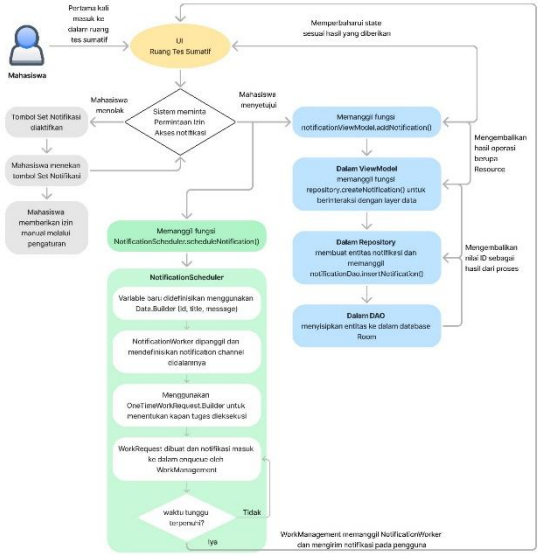
A. Hasil Pembuatan Fitur Notifikasi dengan Penerapan Arsitektur Model-View-ViewModel

Implementasi sistem DIL-MicLearn berbasis website yang dirancang sebelumnya memiliki keterbatasan [3], [5], [6], terutama dalam menampilkan pemberitahuan secara langsung. Mahasiswa harus secara manual masuk ke dalam sistem untuk memperoleh informasi pembelajaran yang relevan, seperti tanggal ujian dan pengumuman lainnya.

Dengan implementasi sistem berbasis mobile, muncul keuntungan yang signifikan. Aplikasi mobile dapat memanfaatkan perangkat smartphone untuk memberikan pemberitahuan secara langsung kepada pengguna tanpa perlu masuk ke dalam sistem. Hal ini memudahkan mahasiswa untuk menerima informasi pembelajaran yang penting secara tepat waktu. Oleh karena itu, fitur pemberitahuan atau notifikasi diimplementasikan dalam sistem DIL-MicLearn berbasis mobile.

Fitur notifikasi dalam sistem DIL-MicLearn diimplementasikan menggunakan arsitektur *Model-View-ViewModel* (MVVM). Fitur notifikasi ini diimplementasikan pada fitur pelaksanaan tes sumatif. Tes sumatif adalah tes atau ujian yang dapat dikerjakan setelah mahasiswa menyelesaikan semua materi dan tes formatif yang tersedia. Tes sumatif ini tidak seperti tes formatif yang dapat dikerjakan kapan pun mahasiswa siap, namun tes ini hanya dapat dikerjakan pada tanggal yang telah diatur oleh pengajar [3], [5]. Jadi mahasiswa dituntut untuk mengingat waktu pengerjaan tes agar dapat mempersiapkan diri dengan baik. Berikut skema alur pengimplementasian notifikasi pada fitur tes sumatif dapat dilihat pada Gambar 7.

Dalam arsitektur MVVM, pemisahan antara logika bisnis dan tampilan antar muka dilakukan menjadi komponen-komponen terpisah, namun tetap berkomunikasi satu sama lain. Hal ini terlihat pada proses pemanggilan fungsi *addNotification* dari tampilan antar muka menuju *ViewModel*. *ViewModel* berinteraksi dengan *repository* melalui fungsi *createNotification*, yang bertugas untuk membuatkan suatu entitas notifikasi baru dengan berbagai informasi yang diperlukan, dan mengirimkannya kepada DAO melalui fungsi *insertNotification*. Selanjutnya, DAO melaksanakan tugasnya untuk menyisipkan data tersebut ke dalam *database* lokal dengan mengelola *query insert*.

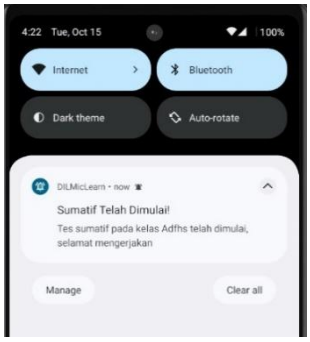


Gambar 7. 1 Skema implementasi notifikasi dalam fitur tes sumatif

Proses yang ditunjukkan oleh kotak berwarna hijau pada Gambar 7 merujuk pada penjadwalan notifikasi yang dilakukan oleh *WorkManager*. Setelah data notifikasi berhasil disimpan dalam database lokal, *WorkManager* akan dipanggil oleh antarmuka pengguna untuk menangani proses penjadwalan melalui pemanggilan fungsi *scheduleNotification* yang terdapat pada komponen *SchedulerNotification*. Fungsi ini akan membuat beberapa variabel, yaitu id notifikasi, *title* sebagai judul notifikasi, dan *message* sebagai isi utama notifikasi.

Selanjutnya, data yang telah dibuat tersebut akan dimasukkan ke dalam variabel bernama *notificationWork* yang dieksekusi satu kali dengan menggunakan fungsi *OneTimeWorkRequest.Builder*. Setelah proses ini, tugas tersebut akan dijadwalkan untuk dieksekusi melalui *enqueue*, sehingga pada waktu yang telah ditentukan, notifikasi akan diaktifkan sesuai jadwal yang telah ditetapkan.

Dengan pengimplementasian skema tersebut, fitur notifikasi dari sisi mahasiswa dalam sistem DIL-MicLearn berbasis *mobile* terlihat seperti Gambar 8.

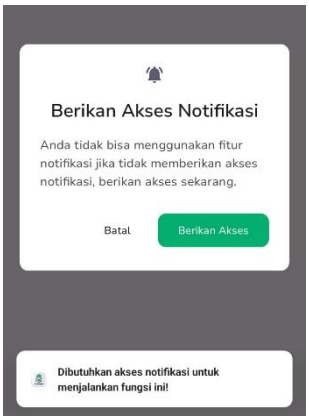


Gambar 8. Hasil implementasi fitur notifikasi

Ketika mahasiswa pertama kali masuk dalam halaman Ruang Tes Sumatif, permintaan pemberian akses notifikasi dikirimkan oleh sistem untuk melanjutkan proses untuk mengaktifkan fitur notifikasi. Jika permintaan akses notifikasi diizinkan oleh mahasiswa, maka tampilan tombol notifikasi secara langsung diperbarui seperti yang ditujukkan pada Gambar 9. Hal ini berarti implemenatasi skema telah berjalan sesuai dengan yang disajikan pada Gambar 7.

Gambar 9. Antarmuka pada halaman Ruang Tes Sumatif dengan pengimplementasian notifikasi

Dalam dikondisi tertentu, mahasiswa dapat menolak permintaan akses notifikasi yang diajukan diawal proses. Namun hal ini membuat mahasiswa harus mengaktifkan secara manual permintaan izin akses notifikasi pada pengaturan aplikasi. Jika mahasiswa tetap memaksa untuk menekan tombol pengingat secara manual, maka sistem merespon dengan memunculkan *pop-up* yang memberi tau agar mahasiswa dapat memberi akses notifikasi secara manual dan mahasiswa akan diarahkan masuk ke pengaturan aplikasi DIL-MicLearn. Kondisi ini dapat dilihat pada Gambar 10.



Gambar 10. Kondisi ketika mahasiswa menolak permintaan akses notifikasi diawal proses

5. KESIMPULAN

Dari hasil penelitian ini dapat disimpulkan bahwa penerapan arsitektur MVVM membuat proses pengembangan aplikasi menjadi lebih terstruktur, dengan pembagian tugas yang jelas di setiap komponennya. Pemisahan logika bisnis dengan antarmuka menghasilkan program yang lebih mudah dipahami, dipelihara, dan dikembangkan oleh programmer lain.

Selain itu, penerapan arsitektur MVVM dalam fitur notifikasi juga membuktikan bahwa arsitektur ini dapat mendukung pengolahan data yang efisien dan reaktif. Model bertugas untuk memproses data yang diambil dari API atau database lokal, repository berfungsi sebagai perantara pengambilan dan penyimpanan data, sedangkan ViewModel bertanggung jawab mengolah data tersebut agar siap ditampilkan ke antarmuka. Dengan pendekatan ini, fitur notifikasi dapat diimplementasikan secara optimal, memungkinkan mahasiswa menerima permberitahuan terkait tes sumatif secara tepat waktu tanpa perlu masuk ke dalam sistem secara manual.

Keuntungan lain dari implementasi sistem berbasis mobile adalah kemudahan pengguna dalam mengakses informasi secara langsung melalui perangkat smartphone. Hal ini mempermudah mahasiswa untuk mengingat jadwal tes sumatif dan mempersiapkan diri

dengan baik. Penjadwalan notifikasi menggunakan WorkManager juga memastikan bahwa pemberitahuan dapat diaktifkan secara terjadwal, sehingga meningkatkan efektivitas fitur yang dikembangkan.

Dengan demikian, penerapan arsitektur MVVM dan fitur notifikasi dapa sistem DIL-MicLearn berbasis *mobile* tidak hanya meningkatkan fungsional aplikasi, tetapi juga memberikan pengalaman yang lebih baik kepada pengguna dan mendukung tujuan pembelajaran yang efektif.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Universitas Pendidikan Ganesha yang telah memberi dukungan terhadap penelitian ini.

DAFTAR PUSTAKA

[1] A. Gormantara, A. L. Tungadi, and R. Y. Carolus, "Pengembangan Sistem E-Learning Berbasis Moodle," *J. Ris. Komputer)*, vol. 10, no. 1, pp. 109–116, 2023, doi: 10.30865/jurikom.v10i1.5527.

[2] S. Fatmawati, "Efektivitas Forum Diskusi Pada E-Learning Berbasis Moodle Untuk Meningkatkan Partisipasi Belajar," *Refleks. Edukatika J. Ilm. Kependidikan*, vol. 9, no. 2, pp. 210–216, 2019, doi: 10.24176/re.v9i2.3379.

[3] N. W. Marti, I. G. P. Suharta, K. Agustini, I. K. Sudarma, I. N. S. W. Wijaya, and L. P. T. Ariani, "Development of The Proposed Microlearning-Based Dynamic Intellectual Learning System to Actualize an Effective Learning Process in Online Environment," *J. Pendidik. Teknol. dan Kejuru.*, vol. 21, no. 1, pp. 57–68, 2024, doi: 10.23887/jptkundiksha.v21i1.75672.

[4] N. W. Marti and L. P. Tuti Ariani, "Pengembangan Konten Pembelajaran Berbasis Micro-Learning Untuk Mata Kuliah Basis Data Di Program Studi S1 Ilmu Komputer-Undiksha [Development of Micro Learning-Based Learning Content for Database Courses in the Computer Science-Undiksha Undergraduate Stud," *J. Pendidik. Tek. dan Kejuru.*, vol. 20, no. 1, pp. 1–12, 2023, [Online]. Available: https://ejournal.undiksha.ac.id/index.php/JPTK/article/view/54572/24658

[5] S. Putu Andi, M. Ni Wayan, Y. Komang Andrew Tri Yoga, A. Dewa Putu Sukra, and S. Kadek Rama, "User Acceptance Test Terhadap Sistem DIL-Miclearn Sebagai E-Learning Pencapaian Ketuntasan Belajar," *Inser. Inf. Syst. Emerg. Technol. J.*, vol. 5, no. 1, pp. 63–72, 2024, doi: 10.23887/insert.v5i1.77859.

[6] N. W. Marti, "Pengembangan Sistem Dynamic Intellectual Learning Berbasis Microlearning Untuk Pencapaian Mastery Learning Pada Mata Kuliah Basis Data Di Program Studi S1 Ilmu Komputer," Pascasarjana, Singaraja, 2023.

[7] N. Rachman and P. Irfan, "Aplikasi Kurir Mobil Pick Up Berbasis Mutli Platform," *JTIM J. Teknol. Inf. dan Multimed.*, vol. 2, no. 3, pp. 166–173, 2020, doi: 10.35746/jtim.v2i3.108.

[8] F. T. Dewantoro and A. F. Waluyo, "Penerapan Rest Api Dalam Perancangan Aplikasi Reservasi Perawatan dan Penitipan Hewan Berbasis Android," *KLIK Kaji. Ilm. Inform. dan Komput.*, vol. 4, no. 2, pp. 1011–1020, 2023, doi: 10.30865/klik.v4i2.1262.

[9] Hendra Utama, Safuan, and Musa Alkhadimi Alhabsy, "Implementasi Aplikasi Penerimaan Mahasiswai Baru Berbasis Android Dengan Fitur Push Notifikasi Program Pascasarjana Universitas Jayabaya," *J. Cakrawala Ilm.*, vol. 1, no. 10, pp. 2387–2396, 2022, doi: 10.53625/jcijurnalcakrawalailmiah.v1i10.2568.

[10] M. S. Arif, A. Musthafa, and D. Muriyatmoko, "Implementation of Model-View-ViewModel (MVVM) Architecture Pattern in the Sistem Informasi Akademik UNIDA Gontor Mobile Application," *Proceeding Int. Conf. Sci. Eng.*, vol. 3, no. April, pp. 283–289, 2020, doi: 10.14421/icse.v3.514.

[11] M. I. Alfathar *et al.*, "Penerapan MVVM (Model View Viewmodel) pada Pengembangan Aplikasi Bank Sampah Digital," *J. Ris. dan Apl. Mhs. Inform.*, vol. 5, no. 2, pp. 406–414, 2024, doi: 10.30998/jrami.v5i2.11071.

[12] B. Arfianto and A. Prapanca, "Analisis Perbandingan Performa Pola Arsitektur Model-View-ViewModel (MVVM) dan

Model-View-Presenter (MVP) pada Pengembangan Aplikasi Desa Wisata Berbasis Android," *J. Informatics Comput. Sci.*, vol. 06, no. Mvvm, pp. 465–478, 2024.

[13] F. P. Sary, A. Prasetio, and M. Moslem, "Analisis Faktor-Faktor Kesuksesan E-Learning dalam Meningkatkan Proses Belajar Mengajar Di Universitas Telkom," *JINOTEP (Jurnal Inov. dan Teknol. Pembelajaran) Kaji. dan Ris. Dalam Teknol. Pembelajaran*, vol. 8, no. 3, pp. 194–206, 2021, doi: 10.17977/um031v8i22021p194.

[14] F. F. Iskandar, G. A. A. Wisudiawan, and S. Y. Puspitasari, "Implementasi dan Evaluasi Pengaruh MVVM Pattern terhadap Reusability pada Aplikasi Berbasis Mobile Android (Studi Kasus Sidang Tugas Akhir S1 Informatika Telkom University)," in *e-Proceeding of Engineering*, 2023, vol. 10, no. 2, pp. 1790–1807.

[15] I. P. R. P. Putra and H. Tolle, "Pengembangan Aplikasi Pembelajaran Bahasa Bali berbasis Android menggunakan MVVM Architecture dan Jetpack Compose," *J. Pengemb. Teknlogi Inf. dan Ilmu Komput.*, vol. 7, no. 5, pp. 2205–2214, 2023, [Online]. Available: https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/12687%0Ahttps://j-ptiik.ub.ac.id/index.php/j-ptiik/article/download/12687/5764

[16] A. F. Diansyah, M. R. Rahman, R. Handayani, D. D. Nur Cahyo, and E. Utami, "Comparative Analysis of Software Development Lifecycle Methods in Software Development: A Systematic Literature Review," *Int. J. Adv. Data Inf. Syst.*, vol. 4, no. 2, pp. 97–106, 2023, doi: 10.25008/ijadis.v4i2.1295.

[17] R. A. J. Arsana, K. Y. E. Aryanto, and N. W. Marti, "Implementasi Sistem Informasi Toko Online pada Toko Lovelace Bali," *JITET (Jurnal Inform. dan Tek. Elektro Ter.*, vol. 14, no. 1, pp. 1575–1582, 2026, doi: https://doi.org/10.23960/jitet.v14i1.9110.

[18] W. Nurhayati, S. Sudarmaji, and G. Yanti Kemala Sari Siregar, "Implementasi Metode Waterfall Pada Sistem Informasi Perpustakaan Online Smk Negeri 1 Seputih Agung," *JIKI (Jurnal Ilmu Komput. Dan Inform.*, vol. 4, no. 2, pp. 196–207, 2023.