BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 5, August 2026 | Hal 1965-1977
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i5.1295

Perancangan dan Implementasi Sistem Microlearning Menggunakan Arsitektur Model-View-ViewModel (MVVM)

Gusti Putu Wulandari*, Jefry Sunupurwa Asri, Nizirwan Anwar, Adi Widiantono

Fakultas Ilmu Komputer, Program Studi Teknik Informatika, Universitas Esa Unggul, Jakarta, Indonesia
Email: [1,*]gustiputuwulandari1409@student.esaunggul.ac.id, [2]jefry.sunupurwa@esaunggul.ac.id, [3]nizirwan.anwar@esaunggul.ac.id, [4]adi.widiantono@esaunggul.ac.id
Email Penulis Korespondensi: gustiputuwulandari1409@student.esaunggul.ac.id

Abstrak−Pengembangan aplikasi mobile *microlearning* memerlukan arsitektur perangkat lunak yang mampu mendukung pengelolaan *state* secara konsisten, pemisahan tanggung jawab antarkomponen, serta kemudahan pemeliharaan aplikasi. Meskipun penelitian mengenai *microlearning* telah banyak membahas aspek pedagogis, kajian yang berfokus pada penerapan arsitektur perangkat lunak, khususnya *Model-View-ViewModel* (MVVM), dalam pengembangan aplikasi pembelajaran berbasis mobile masih relatif terbatas. Penelitian ini bertujuan merancang, mengimplementasikan, dan mengevaluasi sistem mobile *microlearning* menggunakan arsitektur *Model-View-ViewModel* (MVVM). Sistem dikembangkan menggunakan *framework* Flutter pada sisi klien dan Laravel sebagai *backend* yang berkomunikasi melalui REST API dalam arsitektur *client-server*. Penerapan MVVM digunakan untuk memisahkan logika presentasi, pengelolaan *state*, dan antarmuka pengguna sehingga mendukung modularitas, *maintainability*, serta konsistensi aliran data pada aplikasi. Penelitian menggunakan paradigma *Design Science Research* (DSR) dengan metode *Rapid Application Development* (RAD) sebagai pendekatan pengembangan artefak. Hasil pengujian fungsional menggunakan *Black-Box Testing* menunjukkan seluruh skenario pengujian berjalan sesuai dengan hasil yang diharapkan. Evaluasi terhadap sistem pada lingkungan studi kasus juga menunjukkan bahwa sistem mampu mendukung penyajian materi *microlearning*, pengelolaan tugas, dan pelacakan progres belajar secara terintegrasi. Hasil penelitian menunjukkan bahwa penerapan arsitektur MVVM memberikan struktur aplikasi yang lebih modular dan mendukung pengelolaan *state* yang konsisten pada sistem mobile *microlearning*.

Kata Kunci: Microlearning; Model-View-ViewModel (MVVM); Aplikasi Mobile; Flutter; Arsitektur Perangkat Lunak

Abstract−The development of mobile *microlearning* applications requires a software architecture that supports consistent *state* management, separation of concerns among components, and application maintainability. Although previous studies on *microlearning* have largely focused on pedagogical aspects, research addressing software architecture, particularly Model-View-ViewModel (MVVM), in mobile learning applications remains relatively limited. This study aims to design, implement, and evaluate a mobile *microlearning* system using the Model-View-ViewModel (MVVM) architecture. The system is developed using the Flutter framework on the client side and Laravel as the *backend*, communicating through a REST API within a client-server architecture. MVVM is applied to separate presentation logic, *state* management, and the user interface, thereby supporting application modularity, maintainability, and consistent data flow. The study adopts the Design Science Research (DSR) paradigm and Rapid Application Development (RAD) as the software development approach. Functional testing using Black-Box Testing showed that all test scenarios produced the expected results. Evaluation of the artifact within the case study environment also demonstrated that the system supports integrated *microlearning* content delivery, assignment management, and learning progress tracking. The findings indicate that applying the MVVM architecture supports a more modular application structure and consistent *state* management in a mobile *microlearning* system.

Keywords: Microlearning; Model-View-ViewModel (MVVM); Mobile Application; Flutter; Software Architecture

1. PENDAHULUAN

Transformasi digital pada sektor pendidikan telah mendorong pergeseran dari pembelajaran konvensional menuju pembelajaran digital yang lebih fleksibel dan mudah diakses melalui berbagai perangkat. Perkembangan tersebut tidak hanya mengubah cara materi pembelajaran disampaikan, tetapi juga meningkatkan kebutuhan akan sistem perangkat lunak yang mampu mengelola konten, interaksi pengguna, serta pertukaran data secara efektif [1], [2]. Dalam konteks pengembangan aplikasi pembelajaran berbasis mobile, kualitas pengalaman belajar tidak hanya dipengaruhi oleh desain instruksional, tetapi juga oleh arsitektur perangkat lunak yang mendukung modularitas, *maintainability*, dan konsistensi pengelolaan data [3].

Salah satu pendekatan yang banyak diterapkan pada pembelajaran digital adalah *mobile learning*, yaitu penyelenggaraan proses belajar melalui perangkat bergerak yang memungkinkan pengguna mengakses materi secara fleksibel tanpa dibatasi ruang dan waktu [4]. Seiring meningkatnya penggunaan perangkat mobile, pendekatan *microlearning* berkembang sebagai strategi penyajian materi ke dalam unit-unit pembelajaran yang ringkas dan terstruktur sehingga lebih sesuai dengan karakteristik penggunaan perangkat bergerak [5]. Namun, implementasi *microlearning* yang efektif memerlukan dukungan arsitektur perangkat lunak yang mampu mengelola struktur materi, status pembelajaran, serta interaksi pengguna secara konsisten [6], [7].

Dalam pengembangan aplikasi mobile modern, kualitas perangkat lunak tidak hanya ditentukan oleh kelengkapan fitur, tetapi juga oleh kemampuan arsitektur aplikasi dalam memisahkan tanggung jawab antarkomponen, mengelola *state* aplikasi secara konsisten, serta mempermudah proses pengembangan dan pemeliharaan sistem [8]. Salah satu pendekatan arsitektur yang digunakan dalam pengembangan aplikasi mobile adalah *Model-View-ViewModel* (MVVM), yang memisahkan komponen antarmuka dan proses bisnis sehingga mendukung *separation of concerns* serta meningkatkan kemudahan pengembangan dan pemeliharaan aplikasi [8], [9].

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 5, August 2026 | Hal 1965-1977
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i5.1295

Permasalahan tersebut juga ditemukan pada Be The Star Education Learning Center sebagai objek studi kasus penelitian ini. Berdasarkan hasil observasi dan wawancara, akses materi pembelajaran masih bergantung pada jaringan intranet lokal, belum tersedia aplikasi mobile, struktur materi belum tersusun secara modular, serta belum terdapat mekanisme pengelolaan tugas maupun pelacakan progres belajar yang terintegrasi. Kondisi tersebut menunjukkan bahwa sistem yang digunakan belum didukung oleh arsitektur aplikasi yang mampu mengelola aliran data dan *state* aplikasi secara konsisten.

Berbagai penelitian mengenai *microlearning* menunjukkan bahwa pendekatan ini mampu meningkatkan fleksibilitas pembelajaran, retensi pengetahuan, serta keterlibatan pengguna melalui penyajian materi dalam unit-unit yang lebih ringkas [10]. Penelitian lain mengenai *mobile learning* juga menegaskan pentingnya kualitas antarmuka, *usability*, dan kemudahan akses dalam meningkatkan pengalaman belajar pengguna [11]. Di sisi lain, penelitian pada bidang rekayasa perangkat lunak menunjukkan bahwa penerapan arsitektur MVVM memberikan keuntungan berupa pemisahan tanggung jawab yang lebih baik, pengelolaan *state* yang konsisten, serta peningkatan *maintainability* aplikasi [9], [12]. Meskipun demikian, sebagian besar penelitian tersebut masih membahas aspek pedagogi *microlearning* atau mengevaluasi arsitektur MVVM secara umum, sehingga integrasi keduanya dalam pengembangan aplikasi mobile *microlearning* masih relatif terbatas. Meskipun penelitian-penelitian tersebut menunjukkan hasil yang positif, sebagian besar masih mengevaluasi kedua aspek tersebut secara terpisah sehingga belum memberikan gambaran mengenai bagaimana penerapan MVVM dapat mendukung kualitas arsitektur pada sistem mobile *microlearning* secara terpadu.

Berdasarkan kajian tersebut, masih terdapat kesenjangan penelitian pada pengembangan sistem mobile *microlearning* berbasis arsitektur perangkat lunak. Sebagian besar penelitian terdahulu berfokus pada efektivitas pedagogi *microlearning* atau pengembangan fungsionalitas aplikasi, sedangkan penelitian yang secara khusus mengintegrasikan arsitektur MVVM untuk mendukung modularitas aplikasi, *separation of concerns*, pengelolaan *state* yang konsisten, dan *maintainability* pada sistem mobile *microlearning* masih terbatas. Oleh karena itu, diperlukan penelitian yang tidak hanya mengembangkan aplikasi *microlearning*, tetapi juga mengevaluasi kontribusi penerapan MVVM terhadap kualitas arsitektur perangkat lunak.

Berdasarkan kesenjangan tersebut, penelitian ini merancang dan mengimplementasikan sistem mobile *microlearning* menggunakan arsitektur MVVM. Aplikasi dikembangkan menggunakan Flutter sebagai *client* dan Laravel sebagai *backend* yang berkomunikasi melalui REST API dalam arsitektur *client-server*. Penerapan MVVM digunakan untuk memisahkan logika presentasi, pengelolaan *state*, dan antarmuka pengguna sehingga mendukung pengembangan aplikasi yang lebih modular, mudah dipelihara, serta mampu mengelola proses pembelajaran secara konsisten [13].

Penelitian ini memberikan tiga kontribusi utama. Pertama, menerapkan arsitektur MVVM pada sistem mobile *microlearning* untuk mendukung pemisahan tanggung jawab antar lapisan aplikasi, pengelolaan *state* yang konsisten, serta peningkatan modularitas perangkat lunak. Kedua, mengintegrasikan aplikasi Flutter dengan *backend* Laravel melalui REST API dalam arsitektur *client-server* sebagai infrastruktur komunikasi yang mendukung sinkronisasi data dan implementasi layanan pembelajaran pada aplikasi mobile. Ketiga, mengevaluasi sistem yang dikembangkan melalui pengujian fungsional menggunakan *Black-Box Testing* serta evaluasi *usability* menggunakan *System Usability Scale* (SUS) dan efektivitas penyelesaian tugas menggunakan *Task Completion Rate* (TCR) pada lingkungan studi kasus untuk memberikan gambaran mengenai kelayakan implementasi arsitektur yang diusulkan.

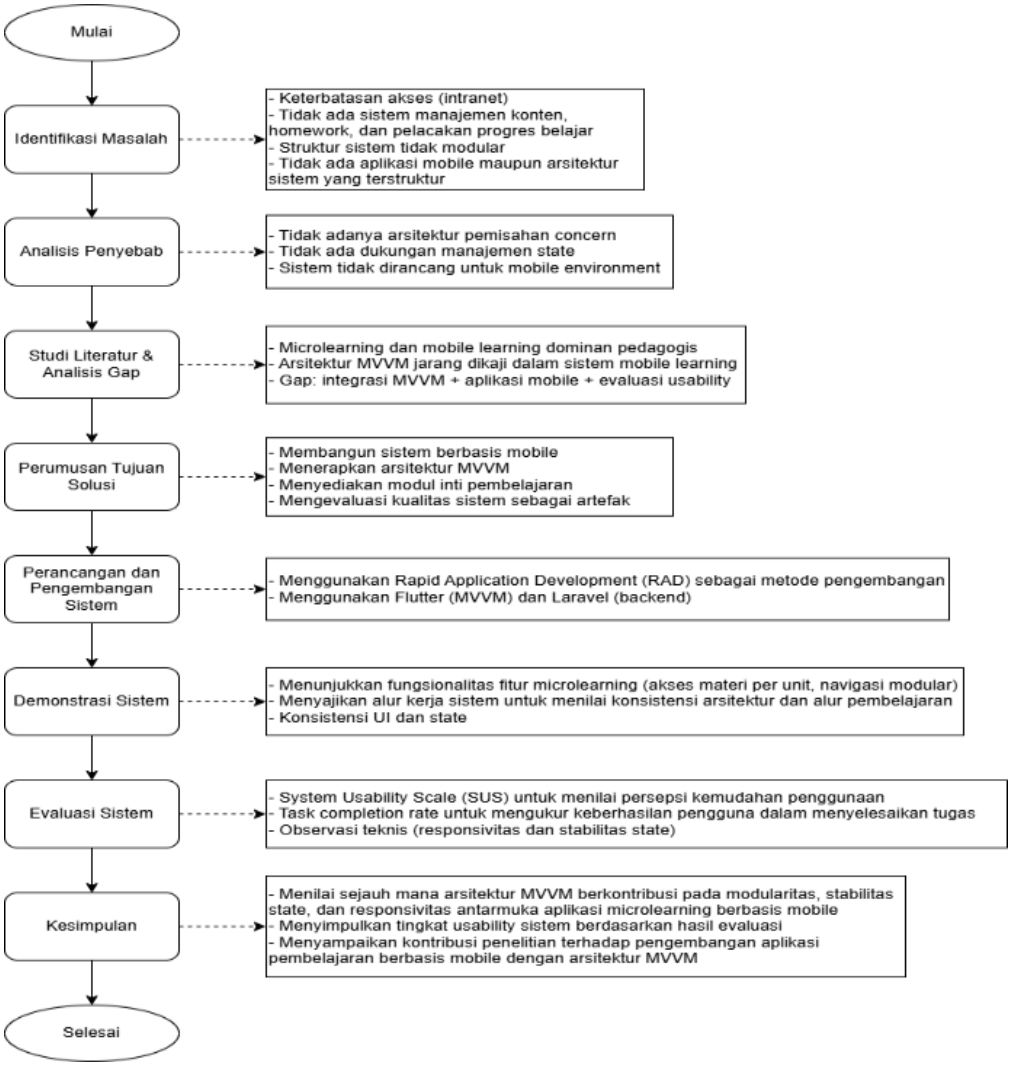
2. METODOLOGI PENELITIAN

2.1 Tahapan *Design Science Research* (DSR)

Penelitian ini menggunakan paradigma *Design Science Research* (DSR) karena tujuan utama penelitian bukan hanya menganalisis suatu fenomena, tetapi merancang, membangun, dan mengevaluasi artefak berupa sistem mobile *microlearning*. DSR dipilih karena menyediakan kerangka kerja yang sistematis untuk mengidentifikasi permasalahan, merumuskan tujuan solusi, mengembangkan artefak, serta mengevaluasi hasil implementasi sehingga sesuai dengan karakter penelitian rekayasa perangkat lunak yang berorientasi pada solusi [14] [15]. Berbeda dengan penelitian yang berorientasi pada pengujian hipotesis, DSR menekankan proses penciptaan artefak sebagai solusi terhadap permasalahan yang telah diidentifikasi [16]. Oleh karena itu, keberhasilan penelitian tidak hanya diukur dari kemampuan artefak untuk diimplementasikan, tetapi juga dari sejauh mana artefak tersebut mampu menjawab kebutuhan pengguna serta memberikan kontribusi terhadap penyelesaian permasalahan pada domain yang diteliti [15].

Pendekatan tersebut memungkinkan proses penelitian dilaksanakan secara iteratif, di mana hasil evaluasi pada setiap tahapan dapat digunakan sebagai masukan untuk penyempurnaan artefak yang dikembangkan. Dengan demikian, pengembangan sistem tidak hanya berorientasi pada penyelesaian kebutuhan fungsional, tetapi juga mempertimbangkan kualitas arsitektur perangkat lunak, keterpeliharaan (*maintainability*), serta kesesuaian solusi terhadap kebutuhan pengguna pada lingkungan studi kasus. Kerangka kerja DSR juga memberikan landasan yang sistematis dalam mendokumentasikan setiap keputusan perancangan sehingga proses pengembangan dapat ditelusuri dan dievaluasi secara lebih terstruktur. Pendekatan ini memastikan bahwa setiap keputusan yang diambil selama proses perancangan dan pengembangan memiliki dasar metodologis yang jelas, sehingga artefak yang dihasilkan tidak hanya memenuhi kebutuhan pengguna, tetapi juga memiliki kualitas rekayasa perangkat lunak yang dapat dipertanggungjawabkan. Oleh karena itu, DSR menjadi landasan utama dalam mengarahkan keseluruhan proses penelitian hingga diperoleh artefak yang siap untuk dievaluasi.

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 5, August 2026 | Hal 1965-1977
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i5.1295



Gambar 1. Alur Kerja Integrasi DSR dan RAD

Berdasarkan alur yang disajikan pada Gambar 1, setiap tahapan dieksekusi secara sekuensial untuk memastikan keterkaitan yang valid antara identifikasi masalah operasional di Be The Star Education Learning Center dengan konstruksi artefak solusi yang diimplementasikan. Penjabaran operasional dari kelima tahapan DSR tersebut dirinci sebagai berikut:

a. Identifikasi Masalah dan Motivasi (*Problem Identification*): Tahap eksplorasi di mana tahap ini berfokus pada dekonstruksi terhadap kendala infrastruktur jaringan lokal (*intranet constraint*) dan fenomena beban kognitif tinggi yang dialami siswa akibat struktur materi instruksional yang monolitik.
b. Definisi Tujuan Solusi (*Objectives of a Solution*): Tahap formulasi di mana tahap ini merumuskan spesifikasi kebutuhan sistem untuk memindahkan gerbang logika bisnis dari manual menjadi terotomatisasi, serta merancang aplikasi mobile yang mampu menyajikan materi modular secara responsif.
c. Desain dan Pengembangan (*Design and Development*): Tahap konstruksi di mana mengonstruksikan artefak berupa sistem *microlearning* dengan mengintegrasikan *backend* Laravel sebagai penyedia layanan data dan aplikasi Flutter sebagai komponen interaksi klien. Tahap ini menaungi fase *User Design* dan *Construction* pada siklus RAD.
d. Demonstrasi dan Evaluasi (*Demonstration and Evaluation*): Tahap validasi di mana sistem yang telah dibangun diuji melalui skenario penggunaan sistem untuk memverifikasi fungsi-fungsi utama aplikasi, seperti pengelolaan *assignment*, *progress tracking*, dan interaksi pengguna. Selanjutnya dilakukan pengukuran empiris terhadap efektivitas penggunaan dan *usability* sistem melalui metrik evaluasi yang telah ditetapkan. Aktivitas ini dieksekusi penuh pada fase *Cutover* dalam metode RAD.
e. Komunikasi (*Communication*): Tahap dokumentasi di mana pada tahap ini menyusun dokumentasi hasil rekayasa sistem ke dalam laporan ilmiah yang siap dipublikasikan untuk memberikan kontribusi pengetahuan bagi komunitas rekayasa perangkat lunak.

2.2 Metode Pengembangan *Rapid Application Development* (RAD)

Untuk mengeksekusi fase implementasi teknis pada tahapan *Design and Development* di dalam kerangka kerja DSR, penelitian ini menerapkan metode *Rapid Application Development* (RAD). Metode ini dipilih karena mendukung proses

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 5, August 2026 | Hal 1965-1977
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i5.1295

pengembangan perangkat lunak secara iteratif melalui pembuatan prototipe dan umpan balik pengguna sehingga kebutuhan sistem dapat divalidasi dan disempurnakan pada setiap siklus pengembangan. Karakteristik tersebut sesuai dengan kebutuhan pembangunan prototipe aplikasi mobile pada penelitian ini [17]. Siklus RAD terdiri atas empat fase utama [18]:

a. Perencanaan Kebutuhan (*Requirement Planning*): Fase ini berfokus pada identifikasi kebutuhan fungsional (*Functional Requirements*/FR) dan non-fungsional (*Non-Functional Requirements*/NFR) dari sistem *microlearning*. Teknik pengumpulan data dilakukan melalui wawancara mendalam terhadap pengajar dan observasi langsung terhadap hambatan interaksi siswa. Keluaran dari fase ini adalah dokumen spesifikasi kebutuhan yang mendefinisikan hak akses pengguna (Admin, Pengajar, Siswa) serta struktur hierarki materi belajar modular.
b. Perancangan Pengguna (*User Design*): Pada fase ini, peneliti dan pengguna bekerja sama secara intensif untuk menerjemahkan kebutuhan fungsional menjadi model arsitektur sistem yang konkret. Proses desain melibatkan pemodelan alur kerja menggunakan *Swimlane Flowchart* usulan, perancangan basis data relasional melalui *Entity Relationship Diagram* (ERD), serta pemodelan aliran data menggunakan *Sequence Diagram*. Selain itu, perancangan antarmuka (UI/UX) dilakukan secara iteratif untuk memastikan navigasi *microlearning* mudah dipahami oleh pengguna. Seluruh keputusan arsitektur, termasuk penerapan arsitektur *client-server* sebagai mekanisme komunikasi dan pola MVVM pada aplikasi mobile, ditetapkan pada fase ini.
c. Konstruksi (*Construction*): Fase konstruksi merupakan tahap di mana seluruh model desain ditranslasikan menjadi kode program fungsional. *Backend engine* dikembangkan menggunakan *framework* Laravel dengan *database* MariaDB, sedangkan aplikasi *mobile client* dikompilasi menggunakan *framework* Flutter (Dart). Sinkronisasi data antara lapisan klien dan *server* lokal dihubungkan melalui protokol RESTful API berbasis pertukaran data JSON. Selain penulisan kode, fase ini juga mencakup pengujian unit internal dan verifikasi validitas fitur secara parsial.
d. Peralihan (*Cutover*): Fase akhir RAD ini berfokus pada pengujian sistem secara menyeluruh di lingkungan operasional dan serah terima sistem. Pada tahap ini, sistem mobile *microlearning* diuji menggunakan metode *Black-Box Testing* untuk memvalidasi fungsi *assignment*, *progress tracking*, autentikasi pengguna, serta fitur utama lainnya. Selanjutnya, pengujian *usability* dilakukan pada kelompok pengguna terfokus (*focused user cohort*) menggunakan metrik *Task Completion Rate* (TCR) dan *System Usability Scale* (SUS) sebelum aplikasi dinyatakan siap digunakan pada lingkungan studi kasus.

DSR dan RAD memiliki peran yang berbeda namun saling melengkapi. DSR digunakan sebagai metodologi penelitian yang mengarahkan keseluruhan proses penelitian mulai dari identifikasi masalah hingga evaluasi sistem. Sementara itu, RAD digunakan sebagai metode pengembangan perangkat lunak pada tahap *Design and Development* dalam DSR untuk merealisasikan sistem yang diusulkan. Dengan demikian, RAD merupakan bagian dari implementasi teknis di dalam kerangka DSR, bukan metodologi penelitian yang berdiri sendiri.

2.3 Pemodelan Sistem

Pemodelan sistem bertindak sebagai jembatan formal untuk menggambarkan struktur statis dan perilaku dinamis dari platform *microlearning* yang dibangun. Pemodelan dirancang secara multi-lapisan untuk memisahkan fokus rekayasa (*separation of concerns*).

a. Pemodelan Proses Bisnis: Pemodelan proses bisnis dilakukan untuk merepresentasikan alur operasional sistem usulan, mulai dari interaksi pengguna hingga mekanisme validasi yang dijalankan oleh sistem. Pemodelan ini digunakan sebagai dasar dalam merancang proses pembelajaran, pengelolaan *assignment*, dan *progress tracking* sehingga seluruh aktivitas pengguna dapat berlangsung secara terstruktur sesuai kebutuhan sistem.
b. Pemodelan Data: Pemodelan data dilakukan untuk mendefinisikan struktur penyimpanan data yang mendukung penyajian materi *microlearning*, *assignment*, dan *progress tracking*. Relasi antartabel dirancang mengikuti kaidah normalisasi hingga Bentuk Normal Ketiga (*Third Normal Form*/3NF) guna meminimalkan redundansi data, menghilangkan ketergantungan transitif, serta menjaga integritas dan konsistensi data selama proses pengelolaan pembelajaran [19].
c. Pemodelan Arsitektur Klien: Pemodelan arsitektur klien dilakukan untuk menggambarkan pembagian tanggung jawab antar komponen *Model*, *View*, dan *ViewModel* pada aplikasi Flutter. Model ini digunakan sebagai acuan implementasi mekanisme *state management*, sehingga logika presentasi terpisah dari antarmuka pengguna serta komunikasi dengan layanan *backend* dilakukan melalui lapisan Model [20].
d. Pemodelan Arsitektur Sistem: Pemodelan arsitektur sistem dilakukan untuk menggambarkan hubungan antar komponen utama yang terdiri atas aplikasi mobile berbasis Flutter, *backend* Laravel, dashboard administrasi, dan basis data dalam topologi *client-server*. Pemodelan ini menjadi acuan implementasi komunikasi antarkomponen melalui REST API serta pembagian tanggung jawab antara lapisan klien dan peladen.

2.4 Metode Pengujian dan Metrik Evaluasi

Evaluasi artefak dilakukan untuk menguji fungsionalitas dan aspek interaksi manusia dengan komputer (HCI). Pengujian dibagi menjadi dua tahapan formal:

a. Validasi Fungsional (*Black-Box Testing*): Pengujian ini dilakukan untuk memverifikasi bahwa seluruh fungsi sistem berjalan sesuai dengan kebutuhan fungsional yang telah ditetapkan tanpa mengevaluasi struktur internal kode program [21]. Pengujian ini dipilih karena penelitian berfokus pada validasi perilaku sistem dari sudut pandang pengguna dan kesesuaian keluaran sistem terhadap skenario yang dirancang. Fokus utama *Black-Box Testing* diarahkan pada

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 5, August 2026 | Hal 1965-1977
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i5.1295

pengujian autentikasi pengguna, pengelolaan struktur *microlearning*, *assignment*, *progress tracking*, serta pembatasan hak akses berbasis peran (*Role-Based Access Control*) pada *backend*.

b. Pengujian Usabilitas (TCR dan SUS): Evaluasi terhadap pengalaman pengguna dilakukan menggunakan metode eksperimen terkontrol pada kelompok pengguna terfokus (*focused user cohort*) yang terdiri dari 6 orang responden dari Be The Star Education Learning Center. Metrik pertama yang diukur adalah *Task Completion Rate* (TCR), yaitu metrik performa *usability* yang mengevaluasi efektivitas pengguna dalam menyelesaikan skenario tugas utama pada aplikasi. Pengukuran ini dipilih karena mampu memberikan gambaran mengenai keberhasilan pengguna dalam menjalankan alur penggunaan sistem yang telah dirancang [22], dengan formula matematika pada persamaan (1):

$$TCR = (\frac{Jumlah\ tugas\ yang\ berhasil\ diselesaikan}{Jumlah\ seluruh\ tugas}) \times 100 \quad (1)$$

Metrik kedua adalah aspek kepuasan subjektif pengguna yang diukur menggunakan instrumen *System Usability Scale* (SUS). SUS merupakan kuesioner standar yang terdiri atas 10 butir pernyataan dengan skala Likert lima poin. Skor setiap responden kemudian dikonversi menjadi nilai 0–100 untuk menginterpretasikan tingkat *usability* dan akseptabilitas sistem berdasarkan kategori interpretasi [23]. SUS digunakan untuk memperoleh gambaran awal mengenai persepsi pengguna terhadap tingkat kemudahan penggunaan aplikasi pada lingkungan studi kasus. Evaluasi ini tidak dimaksudkan untuk menghasilkan generalisasi terhadap populasi yang lebih luas, melainkan sebagai bagian dari evaluasi artefak dalam konteks *Design Science Research*.

2.5 Keterbatasan Evaluasi

Evaluasi sistem dilakukan pada lingkungan Be The Star Education Learning Center sebagai studi kasus penelitian dengan jumlah responden yang terbatas. Oleh karena itu, hasil evaluasi *usability* dan efektivitas penggunaan merepresentasikan kondisi implementasi pada lingkungan tersebut dan belum dimaksudkan untuk digeneralisasikan pada populasi yang lebih luas. Selain itu, penelitian ini belum mencakup pengujian performa sistem pada skenario *multi-user*, pengujian beban (*load testing*), maupun evaluasi *maintainability* menggunakan metrik perangkat lunak secara kuantitatif.

3. HASIL DAN PEMBAHASAN

3.1 Hasil Implementasi Arsitektur *Model-View-ViewModel* (MVVM)

Hasil utama penelitian ini berupa implementasi arsitektur *Model-View-ViewModel* (MVVM) pada sistem mobile *microlearning* yang dikembangkan menggunakan Flutter sebagai aplikasi mobile dan Laravel sebagai *backend*. Penerapan arsitektur ini bertujuan untuk memisahkan tanggung jawab antara antarmuka pengguna, logika presentasi, dan akses data sehingga proses pengembangan maupun pemeliharaan aplikasi menjadi lebih terstruktur. Implementasi tersebut merupakan artefak yang dihasilkan pada tahap *Design and Development* dalam *Design Science Research* (DSR) dengan proses pengembangan perangkat lunak menggunakan metode *Rapid Application Development* (RAD).

Berbeda dengan penelitian yang berfokus pada implementasi fitur aplikasi, penelitian ini menempatkan MVVM sebagai fondasi arsitektur pada sisi aplikasi mobile. Melalui pendekatan ini, komponen *View* hanya bertanggung jawab terhadap penyajian antarmuka pengguna, *ViewModel* mengelola logika presentasi serta *state* aplikasi, sedangkan *Model* menangani komunikasi dengan sumber data melalui lapisan *Service*. Pemisahan tanggung jawab tersebut menerapkan prinsip *separation of concerns* sehingga setiap lapisan dapat dikembangkan, diuji, dan dipelihara secara lebih independen.

Untuk merealisasikan implementasi tersebut, penelitian memanfaatkan beberapa artefak pemodelan yang saling melengkapi, yaitu *Component Diagram*, *Deployment Diagram*, dan *Sequence Diagram*. Ketiga diagram digunakan untuk menggambarkan struktur komponen aplikasi, mekanisme komunikasi data, serta interaksi antar lapisan selama proses pembelajaran berlangsung. Pembahasan setiap diagram disajikan pada subbab berikut.

3.1.1 Implementasi Arsitektur MVVM

Untuk memberikan gambaran mengenai implementasi arsitektur MVVM pada aplikasi mobile, penelitian terlebih dahulu memodelkan hubungan antar komponen penyusun sistem menggunakan *Component Diagram*. Diagram ini menunjukkan pembagian tanggung jawab setiap lapisan aplikasi serta hubungan antar komponen dalam mendukung implementasi prinsip *separation of concerns*.

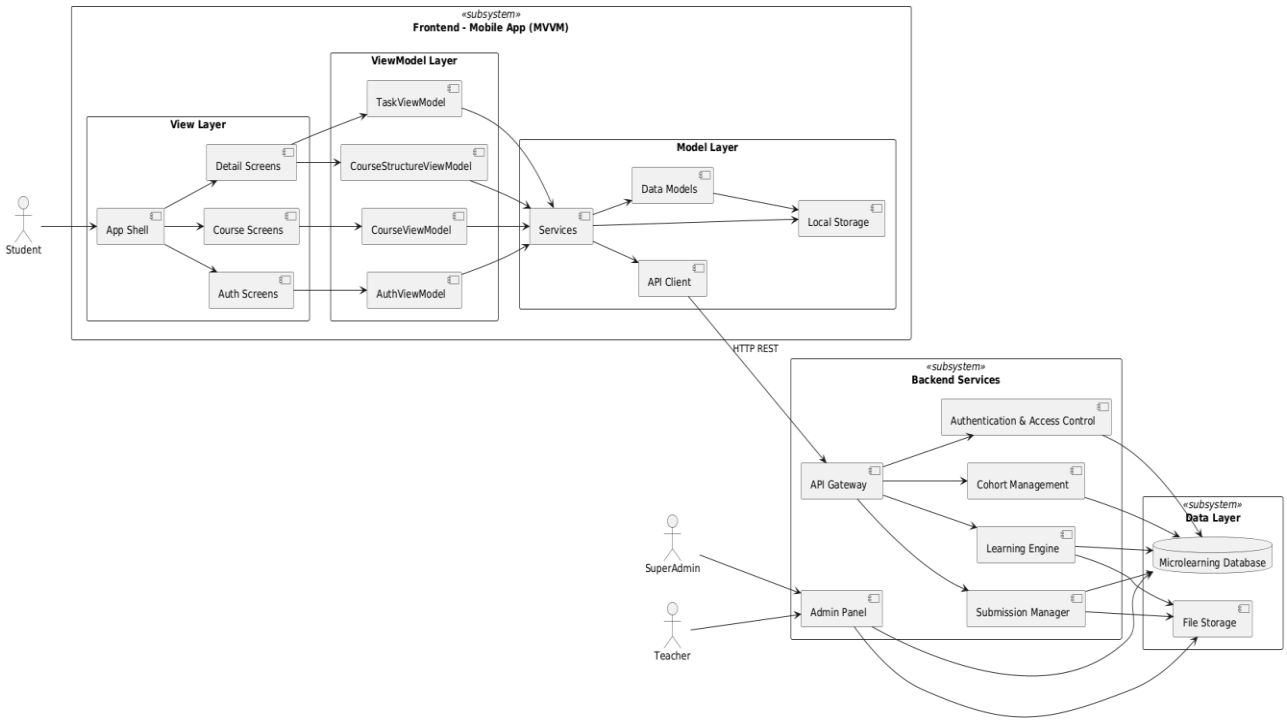
Component Diagram pada Gambar 2 menunjukkan implementasi arsitektur MVVM pada aplikasi mobile. Arsitektur ini membagi aplikasi ke dalam tiga lapisan utama, yaitu *View*, *ViewModel*, dan *Model*, sehingga setiap komponen memiliki tanggung jawab yang berbeda. Pendekatan tersebut bertujuan untuk memisahkan logika presentasi dari antarmuka pengguna serta mengurangi ketergantungan langsung terhadap sumber data.

Lapisan *View* bertanggung jawab menampilkan antarmuka pengguna dan menerima interaksi dari pengguna. Seluruh aksi yang dilakukan pada antarmuka diteruskan kepada *ViewModel* tanpa mengandung logika bisnis. Selanjutnya, *ViewModel* memproses permintaan tersebut, mengelola *state* aplikasi, dan berkomunikasi dengan lapisan *Model* melalui *Service* untuk memperoleh atau memperbarui data. Dengan pendekatan ini, perubahan *state* pada aplikasi dapat direfleksikan secara konsisten pada antarmuka tanpa mencampurkan logika presentasi dengan komponen tampilan.

Lapisan *Model* bertanggung jawab terhadap pengelolaan data dan komunikasi dengan *backend* Laravel melalui REST API. Pada lapisan ini, *Service* berperan sebagai penghubung antara *ViewModel* dan layanan *backend* sehingga

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 5, August 2026 | Hal 1965-1977
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i5.1295

seluruh proses akses data dilakukan secara terpusat. Pendekatan tersebut membuat *ViewModel* tidak bergantung secara langsung pada implementasi layanan *backend* serta meningkatkan modularitas dan *maintainability* aplikasi.

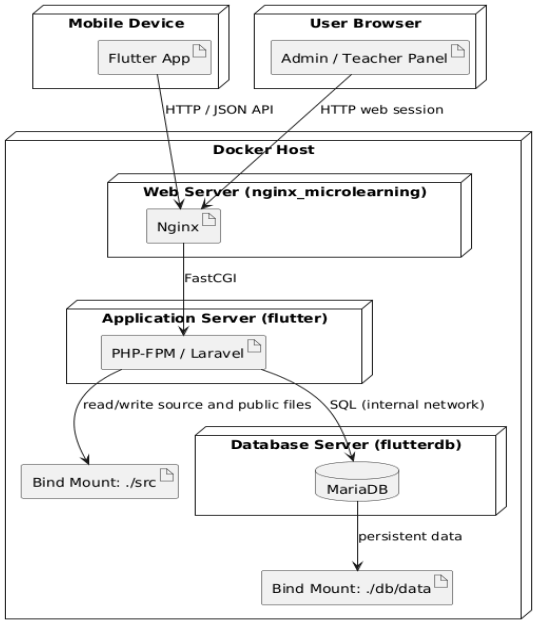


Gambar 2. *Component Diagram*

Berdasarkan implementasi tersebut, penerapan MVVM mendukung prinsip *separation of concerns* melalui pembagian tanggung jawab yang jelas antara *View*, *ViewModel*, dan *Model*. Selain meningkatkan keterbacaan struktur aplikasi, pendekatan ini juga mendukung *maintainability* karena perubahan pada salah satu lapisan dapat dilakukan tanpa memengaruhi lapisan lainnya secara langsung.

3.1.2 Implementasi Komunikasi Data

Setelah implementasi arsitektur MVVM pada aplikasi mobile dijelaskan, langkah berikutnya adalah menggambarkan bagaimana aplikasi berkomunikasi dengan *backend* selama proses pertukaran data. Mekanisme komunikasi tersebut direpresentasikan menggunakan *Deployment Diagram* untuk menunjukkan hubungan antar komponen pada lingkungan implementasi.



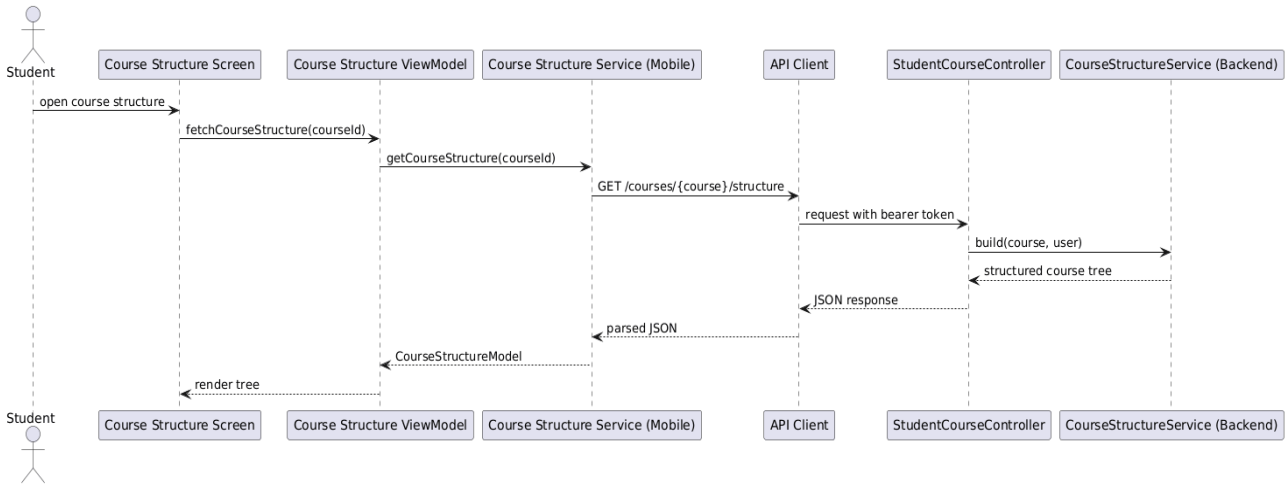
Gambar 3. *Deployment Diagram*

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 5, August 2026 | Hal 1965-1977
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i5.1295

Deployment Diagram pada Gambar 3 menunjukkan lingkungan implementasi sistem mobile *microlearning* yang terdiri atas aplikasi mobile berbasis Flutter, *web server* Nginx, *backend* Laravel, dan basis data MariaDB yang berjalan pada lingkungan Docker. Diagram ini menggambarkan infrastruktur komunikasi antara aplikasi mobile dan *backend* melalui REST API dalam arsitektur *client-server*. Berbeda dengan *Component Diagram* yang menitikberatkan pada implementasi MVVM, *Deployment Diagram* digunakan untuk memperlihatkan lingkungan implementasi dan mekanisme komunikasi antar komponen sistem. Dengan demikian, *Deployment Diagram* berfungsi sebagai representasi lingkungan implementasi sistem, sedangkan implementasi arsitektur MVVM dijelaskan melalui *Component Diagram* dan *Sequence Diagram*.

3.1.3 Implementasi Alur Interaksi Sistem

Untuk memperjelas mekanisme komunikasi antar lapisan aplikasi selama proses pembelajaran berlangsung, penelitian memodelkan alur interaksi menggunakan *Sequence Diagram*.



Gambar 4. *Sequence Diagram* Implementasi MVVM pada Pengambilan Struktur *Course*

Sequence Diagram pada Gambar 4 menunjukkan alur komunikasi antar komponen pada implementasi arsitektur MVVM ketika pengguna mengakses struktur *course*. Proses dimulai dari interaksi pengguna pada *Course Structure Screen* yang diteruskan ke *Course Structure ViewModel*. Selanjutnya, *ViewModel* memanggil *Course Structure Service* pada sisi aplikasi mobile untuk melakukan komunikasi dengan *backend* melalui API Client dan REST API. Permintaan diterima oleh *StudentCourseController*, kemudian diproses lebih lanjut oleh *CourseStructureService* pada sisi *backend* untuk membangun struktur *course* sebelum hasilnya dikembalikan ke aplikasi dalam bentuk respons JSON. Data yang diterima kemudian diproses oleh *ViewModel* dan ditampilkan kembali pada antarmuka pengguna.

3.2 Implementasi Fitur *Microlearning*

Setelah implementasi arsitektur *Model-View-ViewModel* (MVVM) dijelaskan pada subbab sebelumnya, tahap berikutnya adalah merealisasikan arsitektur tersebut ke dalam fitur-fitur utama sistem mobile *microlearning*. Pada penelitian ini, implementasi fitur tidak hanya berfokus pada penyediaan fungsi pembelajaran, tetapi juga pada bagaimana setiap fitur dikembangkan dengan memanfaatkan pembagian tanggung jawab antara *View*, *ViewModel*, dan *Model*. Pendekatan tersebut memungkinkan antarmuka pengguna tetap terpisah dari logika presentasi maupun mekanisme akses data sehingga aplikasi lebih mudah dikembangkan dan dipelihara.

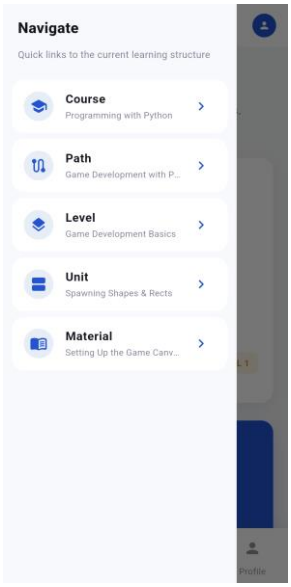
Seluruh data pembelajaran diperoleh melalui backend Laravel menggunakan REST API, kemudian diproses oleh lapisan *Model* dan *ViewModel* sebelum ditampilkan pada antarmuka pengguna. Dengan pendekatan ini, setiap perubahan data dapat direfleksikan secara konsisten pada aplikasi tanpa menempatkan logika bisnis secara langsung pada lapisan View. Implementasi tersebut diterapkan pada seluruh fitur utama sistem, meliputi struktur *microlearning*, mekanisme *assignment*, dan *progress tracking.*

3.2.1 Implementasi Struktur *Microlearning*

Implementasi struktur *microlearning* diawali dengan penyusunan antarmuka navigasi yang memungkinkan pengguna mengakses materi pembelajaran secara bertahap sesuai struktur kurikulum yang telah ditentukan. Tampilan utama aplikasi beserta halaman *detail course* ditunjukkan pada Gambar 5 dan Gambar 6.

Struktur *microlearning* diimplementasikan melalui hierarki pembelajaran yang terdiri atas *Course*, *Path*, *Level*, *Unit*, *Material*, dan *Assignment*. Penyusunan materi secara bertingkat memungkinkan setiap konten pembelajaran disajikan dalam unit-unit yang lebih kecil sehingga pengguna dapat mempelajari materi secara bertahap sesuai urutan yang telah ditentukan. Pendekatan tersebut sejalan dengan karakteristik *microlearning* yang menekankan penyampaian materi secara ringkas dan terstruktur.

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 5, August 2026 | Hal 1965-1977
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i5.1295

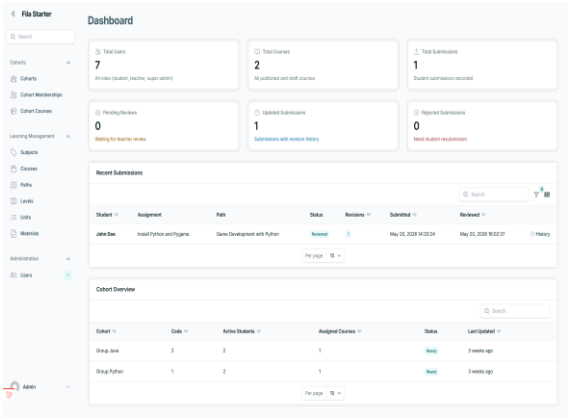


Gambar 5. Navigation Bar

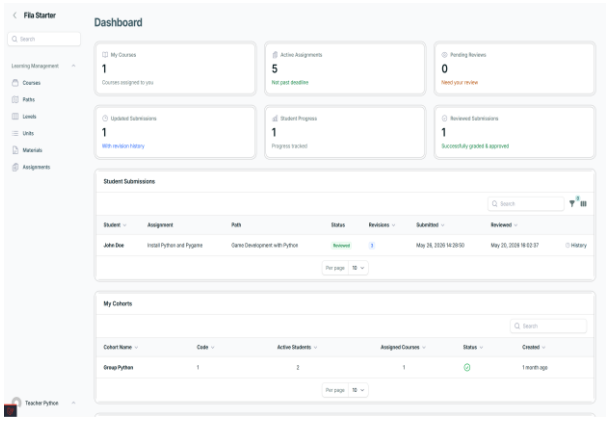
Gambar 6. Halaman Course Detail

Pada aplikasi mobile, struktur pembelajaran ditampilkan melalui lapisan *View*, sedangkan proses pengambilan data dilakukan oleh *ViewModel* melalui *Service* pada lapisan *Model*. Dengan demikian, *View* hanya bertanggung jawab terhadap penyajian informasi, sementara seluruh proses akses data dilakukan secara terpisah sesuai prinsip *separation of concerns*. Pendekatan ini membuat perubahan struktur data pada *backend* tidak memerlukan perubahan langsung pada komponen antarmuka.

Pengelolaan struktur pembelajaran dilakukan melalui dashboard administrator dan pengajar sebagaimana ditunjukkan pada Gambar 7 dan Gambar 8.



Gambar 6. Tampilan Dashboard Admin



Gambar 8. Tampilan Dashboard Pengajar

Pada sisi *backend*, seluruh struktur pembelajaran dikelola melalui dashboard administrator dan pengajar. Setiap perubahan terhadap *Course*, *Path*, *Level*, *Unit*, *Material*, maupun *Assignment* disimpan pada basis data dan disediakan sebagai layanan melalui REST API. Selanjutnya aplikasi mobile memperoleh data tersebut melalui lapisan *Model* sehingga informasi yang ditampilkan kepada pengguna selalu mengacu pada data yang tersimpan pada *backend*.

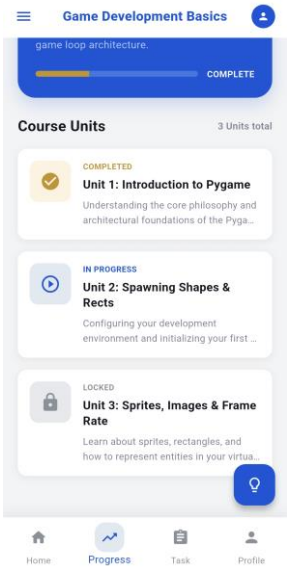
Implementasi tersebut menunjukkan bahwa pemisahan tanggung jawab antara antarmuka pengguna, logika presentasi, dan sumber data mendukung penyajian struktur *microlearning* secara lebih modular. Selain mempermudah pengelolaan materi, pendekatan ini juga meningkatkan konsistensi informasi yang diterima pengguna pada seluruh perangkat.

3.2.2 Implementasi *Assignment*

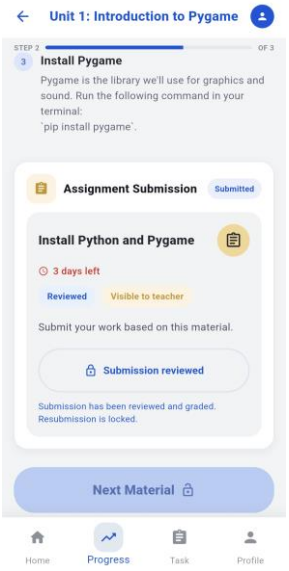
Salah satu fitur utama pada sistem adalah mekanisme *assignment* yang digunakan sebagai bagian dari proses pembelajaran. Implementasi fitur ini memungkinkan pengguna mengakses, mengerjakan, dan mengirimkan tugas melalui aplikasi mobile sebagaimana ditunjukkan pada Gambar 9 dan Gambar 10.

Assignment diimplementasikan sebagai mekanisme yang mendukung proses evaluasi pembelajaran pada setiap unit materi. Setelah pengguna menyelesaikan materi tertentu, aplikasi menyediakan halaman tugas yang memungkinkan pengguna mengirimkan hasil pekerjaan melalui aplikasi mobile. Seluruh proses pengiriman data dilakukan melalui REST API menuju *backend* Laravel untuk diproses dan disimpan.

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 5, August 2026 | Hal 1965-1977
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i5.1295



Gambar 7. Halaman *Unit List*



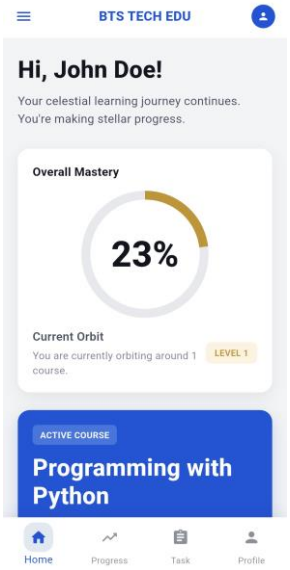
Gambar 8. Halaman Detail Materi dan Penugasan

Pada sisi aplikasi, *View* hanya menampilkan informasi tugas dan menerima interaksi pengguna, sedangkan *ViewModel* bertanggung jawab mengelola proses pengiriman data dan memperbarui *state* aplikasi berdasarkan respons dari *backend*. Pemisahan tanggung jawab tersebut membuat proses pengelolaan *assignment* tidak bergantung pada komponen antarmuka sehingga lebih mudah dipelihara apabila terjadi perubahan aturan maupun layanan *backend*.

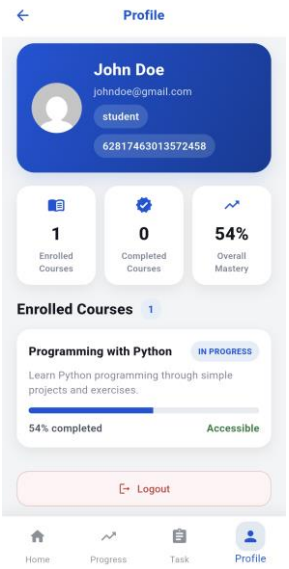
Dengan pendekatan tersebut, implementasi *assignment* tidak hanya menyediakan fungsi pengumpulan tugas, tetapi juga menunjukkan bagaimana arsitektur MVVM mendukung pengelolaan alur interaksi pengguna secara lebih terstruktur melalui pemisahan antara antarmuka, logika presentasi, dan akses data.

3.2.3 Implementasi *Progress Tracking*

Selain menyediakan fitur *assignment*, sistem juga mengimplementasikan mekanisme *progress tracking* untuk membantu pengguna memantau perkembangan proses belajar. Tampilan halaman utama dan informasi progres pengguna ditunjukkan pada Gambar 11 dan Gambar 12.



Gambar 9. *Home Screen* Aplikasi



Gambar 10. Halaman Profil dan Progres Siswa

Progress tracking digunakan untuk menampilkan status penyelesaian aktivitas pembelajaran berdasarkan data yang tersimpan pada *backend*. Informasi tersebut meliputi materi yang telah diselesaikan, progres pembelajaran, serta status aktivitas pengguna selama mengikuti *course*. Seluruh data diperoleh melalui REST API dan diproses oleh lapisan *Model* sebelum diteruskan kepada *ViewModel* untuk memperbarui *state* aplikasi.

Melalui penerapan MVVM, perubahan progres belajar yang diterima dari *backend* dapat langsung direfleksikan pada antarmuka pengguna tanpa memerlukan manipulasi data secara langsung pada lapisan *View*. Pendekatan tersebut menghasilkan aliran data yang lebih konsisten sekaligus mempermudah pengelolaan *state* ketika pengguna berpindah

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 5, August 2026 | Hal 1965-1977
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i5.1295

antarhalaman atau melakukan pembaruan informasi. Pengelolaan *state* melalui *ViewModel* juga membantu menjaga sinkronisasi antara data yang diterima dari *backend* dengan informasi yang ditampilkan pada antarmuka pengguna. Dengan demikian, setiap perubahan progres dapat diperbarui secara konsisten tanpa memerlukan pengelolaan *state* secara langsung pada komponen *View*, sehingga mengurangi potensi inkonsistensi tampilan selama proses pembelajaran berlangsung.

Implementasi progress tracking menunjukkan bahwa penerapan MVVM tidak hanya mendukung penyajian antarmuka yang responsif, tetapi juga membantu menjaga konsistensi pengelolaan *state* aplikasi melalui pemisahan tanggung jawab yang jelas antara *View*, *ViewModel*, dan *Model*.

3.3 Validasi dan Evaluasi Sistem

Setelah implementasi arsitektur *Model-View-ViewModel* (MVVM) beserta fitur-fitur utama sistem selesai dilakukan, tahap berikutnya adalah melakukan validasi dan evaluasi terhadap artefak yang dikembangkan. Pada penelitian *Design Science Research* (DSR), proses evaluasi bertujuan untuk memastikan bahwa artefak mampu memenuhi kebutuhan yang telah ditetapkan serta memberikan solusi terhadap permasalahan yang diidentifikasi pada tahap awal penelitian.

Validasi dilakukan menggunakan *Black-Box Testing* untuk memastikan bahwa seluruh fungsi sistem berjalan sesuai kebutuhan fungsional. Selanjutnya, evaluasi terhadap pengguna dilakukan menggunakan *Task Completion Rate* (TCR) untuk mengukur efektivitas penyelesaian tugas serta *System Usability Scale* (SUS) untuk memperoleh gambaran mengenai persepsi pengguna terhadap kemudahan penggunaan aplikasi pada lingkungan studi kasus.

3.3.1 Validasi Fungsional Sistem

Untuk memastikan bahwa seluruh kebutuhan fungsional telah direalisasikan sesuai dengan spesifikasi sistem, dilakukan pengujian menggunakan metode *Black-Box Testing* terhadap setiap fungsi utama aplikasi. Pengujian meliputi proses autentikasi, pengelolaan struktur *microlearning*, assignment, serta progress tracking. Hasil pengujian ditunjukkan pada Tabel 1.

Tabel 1. Pengujian Sistem dengan *Black-Box Testing*

No.	Skenario Uji	Langkah Pengujian	Hasil yang Diharapkan	Hasil Aktual	Status
1	*Login* dengan kredensial valid	Masukkan email dan *password* yang benar, lalu tekan tombol *login*	Sistem menampilkan halaman utama dan menghasilkan token autentikasi	Sistem menampilkan halaman utama dan menghasilkan token autentikasi	Valid
2	*Login* dengan kredensial tidak valid	Masukkan email atau *password* yang salah	Sistem menampilkan pesan kesalahan *login*	Sistem menampilkan pesan kesalahan *login*	Valid
3	Perubahan kata sandi saat *first login*	*Login* dengan akun yang memiliki status *first login*	Sistem mengarahkan pengguna ke halaman ubah *password*	Sistem mengarahkan pengguna ke halaman ubah *password*	Valid
4	*Logout* pengguna	Tekan tombol *logout* pada aplikasi	Sistem menghapus sesi aktif dan kembali ke halaman *login*	Sistem menghapus sesi aktif dan kembali ke halaman *login*	Valid
5	Menampilkan daftar *course* aktif	Buka halaman *Home*	Sistem menampilkan daftar *course* yang sedang diikuti beserta progresnya	Sistem menampilkan daftar *course* yang sedang diikuti beserta progresnya	Valid
6	Membuka struktur *course*	Pilih salah satu *course* pada daftar *course* aktif	Sistem menampilkan struktur *course* secara hierarkis sesuai akses pengguna	Sistem menampilkan struktur *course* secara hierarkis sesuai akses pengguna	Valid
7	Membuka materi yang terkunci	Pilih materi dengan status *unlocked*	Sistem menolak akses dan tetap menjaga status materi terkunci	Sistem menolak akses dan tetap menjaga status materi terkunci	Valid
8	Membuka materi yang terbuka	Pilih materi dengan status *unlocked*	Sistem menampilkan detail materi secara lengkap	Sistem menampilkan detail materi secara lengkap	Valid
9	Menandai materi selesai	Tekan aksi *next material* pada halaman detail materi	Sistem memperbarui status progres materi menjadi selesai dan berlanjut ke halaman detail materi berikutnya	Sistem memperbarui status progres materi menjadi selesai dan berlanjut ke halaman detail materi berikutnya	Valid

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 6, No 5, August 2026 | Hal 1965-1977
https://hostjournals.com/bulletincsr
DOI: 10.47065/bulletincsr.v6i5.1295

No.	Skenario Uji	Langkah Pengujian	Hasil yang Diharapkan	Hasil Aktual	Status
10	Mengirim *submission* tugas	Unggah file atau isi tugas lalu tekan *submit*	Sistem menyimpan *submission* dan menampilkan status berhasil	Sistem menyimpan *submission* dan menampilkan status berhasil	Valid
11	Mengakses *submission* tanpa memilih file	Buka halaman tugas tanpa memilih file unggahan	Sistem hanya menampilkan opsi *upload file* dan tombol *submit* belum dapat digunakan	Sistem hanya menampilkan opsi *upload file* dan tombol *submit* belum dapat digunakan	Valid
12	Melihat *submission* yang sudah dikirim	Buka halaman *Task*	Sistem menampilkan status *submission*, nilai, dan *feedback* bila tersedia	Sistem menampilkan status *submission*, nilai, dan *feedback* bila tersedia	Valid

Berdasarkan hasil pengujian pada Tabel 1, seluruh skenario menghasilkan keluaran yang sesuai dengan hasil yang diharapkan sehingga seluruh fungsi sistem dinyatakan valid. Hasil tersebut menunjukkan bahwa artefak yang dikembangkan telah memenuhi kebutuhan fungsional yang dirumuskan pada tahap analisis dan mampu menjalankan proses pembelajaran sesuai rancangan.

Keberhasilan seluruh skenario pengujian juga menunjukkan bahwa penerapan arsitektur MVVM mampu mendukung implementasi fitur-fitur utama tanpa mengganggu konsistensi perilaku sistem. Melalui pemisahan tanggung jawab antara *View*, *ViewModel*, dan *Model*, setiap proses interaksi pengguna dapat diproses secara terstruktur sebelum diteruskan menuju *backend* melalui REST API.

3.3.2 Evaluasi Efektivitas dan *Usability*

Selain validasi terhadap fungsi sistem, penelitian juga melakukan evaluasi terhadap efektivitas penggunaan dan *usability* aplikasi. Evaluasi dilakukan menggunakan metrik *Task Completion Rate* (TCR) dan *System Usability Scale* (SUS) untuk memperoleh gambaran mengenai kemampuan pengguna dalam menyelesaikan tugas serta persepsi mereka terhadap kemudahan penggunaan sistem. Ringkasan hasil evaluasi disajikan pada Tabel 2.

Tabel 2. Rekapitulasi Hasil Validasi dan Evaluasi Sistem

Metode Evaluasi	Hasil
Black-Box Testing	Seluruh skenario valid
Task Completion Rate (TCR)	100%
System Usability Scale (SUS)	82,08 (*Grade* A)

Berdasarkan hasil evaluasi pada Tabel 2, seluruh responden berhasil menyelesaikan seluruh skenario tugas yang diberikan sehingga diperoleh nilai *Task Completion Rate* (TCR) sebesar 100%. Hasil tersebut menunjukkan bahwa alur interaksi yang dirancang pada aplikasi dapat dipahami dan dijalankan oleh pengguna sesuai dengan skenario pengujian pada lingkungan studi kasus.

Selanjutnya, pengukuran menggunakan *System Usability Scale* (SUS) menghasilkan skor rata-rata 82,08, yang termasuk dalam kategori *usability* yang baik. Nilai tersebut menunjukkan bahwa pengguna memberikan persepsi positif terhadap kemudahan penggunaan aplikasi selama proses pembelajaran.

Meskipun demikian, hasil evaluasi ini merepresentasikan implementasi sistem pada lingkungan Be The Star Education Learning Center dengan jumlah responden yang terbatas. Oleh karena itu, hasil pengukuran tidak dimaksudkan untuk digeneralisasikan terhadap populasi yang lebih luas, melainkan sebagai bagian dari evaluasi sistem dalam konteks *Design Science Research*. Hasil evaluasi menunjukkan bahwa sistem yang dikembangkan berhasil mendukung implementasi sistem mobile *microlearning* dengan tingkat efektivitas penggunaan dan *usability* yang baik pada lingkungan studi kasus. Temuan tersebut memberikan indikasi bahwa penerapan arsitektur MVVM mampu mendukung penyajian antarmuka yang responsif, pengelolaan *state* yang konsisten, serta pemisahan tanggung jawab antara komponen antarmuka, logika presentasi, dan akses data.

3.4 Pembahasan

Hasil penelitian menunjukkan bahwa penerapan arsitektur *Model-View-ViewModel* (MVVM) pada sistem mobile *microlearning* mampu mendukung pengembangan aplikasi yang lebih modular melalui pemisahan tanggung jawab antara *View*, *ViewModel*, dan *Model*. Pembagian tanggung jawab tersebut memungkinkan setiap lapisan menjalankan fungsi yang spesifik, di mana *View* bertanggung jawab terhadap antarmuka pengguna, *ViewModel* mengelola logika presentasi dan state aplikasi, sedangkan *Model* menangani akses data serta komunikasi dengan *backend* melalui REST API. Pendekatan ini meningkatkan modularitas dan *maintainability* aplikasi karena perubahan pada salah satu lapisan tidak memberikan dampak langsung terhadap lapisan lainnya. Dengan demikian, proses pengembangan maupun pemeliharaan aplikasi dapat dilakukan secara lebih fleksibel tanpa memengaruhi keseluruhan struktur sistem.

Selain meningkatkan modularitas, penerapan MVVM juga memperkuat prinsip *separation of concerns* melalui pemisahan yang jelas antara antarmuka pengguna, logika presentasi, dan akses data. Seluruh proses komunikasi dengan *backend* dilakukan melalui lapisan *Model* dan *ViewModel* sehingga menghasilkan alur pertukaran data yang lebih

terstruktur serta mengurangi ketergantungan antara komponen antarmuka dan mekanisme akses data. Implementasi tersebut diterapkan pada seluruh fitur utama sistem, termasuk penyajian materi *microlearning*, pengelolaan *assignment*, dan *progress tracking*. Pengelolaan *state* melalui *ViewModel* juga memungkinkan setiap perubahan data yang diterima dari *backend* diproses terlebih dahulu sebelum ditampilkan pada antarmuka pengguna, sehingga penyajian informasi menjadi lebih konsisten sekaligus meningkatkan keterbacaan kode program ketika aplikasi dikembangkan lebih lanjut.

Hasil evaluasi menunjukkan bahwa sistem telah memenuhi seluruh kebutuhan fungsional berdasarkan skenario *Black-Box Testing*. Selain itu, nilai *Task Completion Rate* (TCR) sebesar 100% menunjukkan bahwa seluruh responden mampu menyelesaikan skenario tugas yang diberikan, sedangkan skor *System Usability Scale* (SUS) sebesar 82,08 mengindikasikan bahwa aplikasi memiliki tingkat *usability* yang baik pada lingkungan studi kasus. Temuan tersebut menunjukkan bahwa penerapan MVVM tidak hanya memberikan manfaat terhadap kualitas struktur perangkat lunak, tetapi juga mendukung pengembangan aplikasi yang dapat digunakan secara efektif oleh pengguna.

Temuan penelitian ini konsisten dengan penelitian sebelumnya mengenai penerapan MVVM pada sistem mobile *e-learning*, yang menunjukkan bahwa pemisahan antara antarmuka pengguna, logika presentasi, dan akses data mampu menghasilkan struktur aplikasi yang lebih modular serta meningkatkan efisiensi pengembangan aplikasi [24]. Hasil yang serupa juga dilaporkan pada penelitian mengenai pengembangan aplikasi Learning Management System berbasis Flutter dan REST API, di mana penerapan MVVM mempermudah pengembangan dan pemeliharaan aplikasi melalui pemisahan logika presentasi dari antarmuka pengguna [25]. Selain itu, penelitian mengenai penerapan pola *state management* pada Flutter menunjukkan bahwa pengelolaan *state* yang terstruktur mampu meningkatkan *maintainability* melalui penerapan *low coupling* dan *separation of concerns* [26]. Temuan-temuan tersebut memperkuat hasil penelitian ini bahwa penerapan MVVM memberikan manfaat terhadap modularitas, pengelolaan *state*, dan kemudahan pemeliharaan aplikasi.

Meskipun memiliki kesamaan dengan penelitian-penelitian terdahulu, penelitian ini memberikan kontribusi yang berbeda. Sebagian besar penelitian sebelumnya berfokus pada implementasi MVVM pada aplikasi pembelajaran atau evaluasi pola *state management* secara umum. Sementara itu, penelitian ini menerapkan MVVM secara spesifik pada sistem mobile *microlearning* yang terintegrasi dengan *backend* Laravel melalui REST API serta mengevaluasi implikasi penerapan arsitektur tersebut terhadap *modularity*, *state management*, *separation of concerns*, dan *maintainability*. Dengan demikian, penelitian ini tidak hanya menunjukkan implementasi MVVM pada aplikasi mobile, tetapi juga memberikan pembahasan mengenai bagaimana penerapan arsitektur tersebut mendukung kualitas perangkat lunak pada konteks pembelajaran digital berbasis *microlearning*.

Meskipun memberikan berbagai keuntungan dari sisi modularitas dan *maintainability*, penerapan MVVM juga meningkatkan kompleksitas struktur aplikasi. Implementasi arsitektur ini memerlukan pemisahan komponen ke dalam beberapa lapisan, penambahan kelas seperti *ViewModel* dan *Service*, serta pengelolaan alur komunikasi data yang lebih terstruktur dibandingkan pendekatan tanpa pola arsitektur. Kompleksitas tersebut dapat meningkatkan upaya pengembangan pada tahap awal, namun memberikan keuntungan dalam jangka panjang karena aplikasi menjadi lebih mudah dipelihara, dikembangkan, dan diskalakan seiring bertambahnya fitur maupun kebutuhan sistem. Oleh karena itu, pemilihan arsitektur MVVM perlu mempertimbangkan tingkat kompleksitas aplikasi agar manfaat modularitas dan maintainability yang diperoleh sebanding dengan kompleksitas implementasinya.

4. KESIMPULAN

Penelitian ini berhasil merancang dan mengimplementasikan sistem mobile *microlearning* menggunakan arsitektur *Model-View-ViewModel* (MVVM) pada aplikasi berbasis Flutter dengan *backend* Laravel yang berkomunikasi melalui REST API. Penerapan MVVM mampu memisahkan tanggung jawab antara *View*, *ViewModel*, dan *Model*, sehingga mendukung penerapan prinsip *separation of concerns*, pengelolaan *state* yang lebih konsisten, serta meningkatkan modularitas dan *maintainability* aplikasi. Hasil validasi menunjukkan bahwa seluruh skenario *Black-Box Testing* berjalan sesuai dengan kebutuhan fungsional, sedangkan hasil evaluasi memperoleh *Task Completion Rate* (TCR) sebesar 100% dan skor *System Usability Scale* (SUS) sebesar 82,08, yang menunjukkan bahwa aplikasi dapat digunakan secara efektif pada lingkungan studi kasus. Penelitian ini memberikan kontribusi berupa implementasi arsitektur MVVM pada sistem mobile *microlearning* sebagai pendekatan untuk meningkatkan kualitas struktur perangkat lunak tanpa mengubah mekanisme pembelajaran yang diterapkan. Meskipun demikian, penelitian ini masih terbatas pada satu lingkungan studi kasus dengan jumlah responden yang terbatas serta belum mengevaluasi aspek performa sistem maupun *maintainability* secara kuantitatif. Oleh karena itu, penelitian selanjutnya dapat memperluas implementasi pada konteks yang lebih beragam serta menambahkan evaluasi terhadap performa, skalabilitas, maupun metrik kualitas perangkat lunak lainnya.

REFERENCES

[1] Y. Zou, F. Kuek, W. Feng, and X. Cheng, "Digital learning in the 21st century: trends, challenges, and innovations in technology integration," *Front. Educ.*, vol. 10, Mar. 2025, doi: 10.3389/feduc.2025.1562391.

[2] S. Timotheou *et al.*, "Impacts of digital technologies on education and factors influencing schools' digital capacity and transformation: A literature review," *Educ. Inf. Technol.*, vol. 28, no. 6, pp. 6695–6726, Jun. 2023, doi: 10.1007/s10639-022-11431-8.

[3] A. David, D. Mihai, M. E. Mihailescu, M. Carabas, and N. Tapus, "Scalability through Distributed Deployment for Moodle Learning Management System," *Procedia Comput. Sci.*, vol. 214, no. C, pp. 34–41, 2022, doi: 10.1016/j.procs.2022.11.145.

[4] S. Sophonhiranrak, "Features, barriers, and influencing factors of mobile learning in higher education: A systematic review,"

Heliyon, vol. 7, no. 4, p. e06696, Apr. 2021, doi: 10.1016/j.heliyon.2021.e06696.

[5] W. K. Monib, A. Qazi, and R. A. Apong, "Microlearning beyond boundaries: A systematic review and a novel framework for improving learning outcomes," *Heliyon*, vol. 11, no. 2, p. e41413, Jan. 2025, doi: 10.1016/j.heliyon.2024.e41413.

[6] Rahmaniar, Sapto Haryoko, Muhammad Rais, and Supriadi, "LAMACCA: Learning Analytics–Driven Adaptive Modular LMS Architecture for Collaborative Cyber Classrooms in Higher Education," *J. Vocat. Informatics Comput. Educ.*, pp. 151–166, Dec. 2025, doi: 10.66053/voice.v3i2.468.

[7] R. Lv, Q. Liu, W. Gao, H. Zhang, J. Lu, and L. Zhu, "GenAL: Generative Agent for Adaptive Learning," *Proc. AAAI Conf. Artif. Intell.*, vol. 39, no. 1, pp. 577–585, Apr. 2025, doi: 10.1609/aaai.v39i1.32038.

[8] I. M. Riyadhi, Intan Purnamasari, and Kamal Prihandani, "Penerapan Pola Arsitektur Mvvm Pada Perancangan Aplikasi Pengaduan Masyarakat Berbasis Android," *INFOTECH J.*, vol. 9, no. 1, pp. 147–158, May 2023, doi: 10.31949/infotech.v9i1.5246.

[9] F.-E. Suarez-Carvajal, R. Lobo-Quintero, J. Santoyo, J. Pinzon-Castellanos, and Y. Gamba-Gonzalez, "MVVM in the Era of Modern Android: A Systematic Literature Review and Taxonomy of Architectural Trade-offs," *CLEI Electron. J.*, vol. 29, no. 2, Jul. 2026, doi: 10.19153/cleiej.29.2.8.

[10] A. D. Samala, L. Bojic, D. Bekiroğlu, R. Watrianthos, and Y. Hendriyani, "Microlearning: Transforming Education with Bite-Sized Learning on the Go—Insights and Applications," *Int. J. Interact. Mob. Technol.*, vol. 17, no. 21, pp. 4–24, Nov. 2023, doi: 10.3991/ijim.v17i21.42951.

[11] M. Ahmad Faudzi, Z. Che Cob, R. Omar, S. A. Sharudin, and M. Ghazali, "Investigating the User Interface Design Frameworks of Current Mobile Learning Applications: A Systematic Review," *Educ. Sci.*, vol. 13, no. 1, p. 94, Jan. 2023, doi: 10.3390/educsci13010094.

[12] D. Indrawan, D. S. Kusumo, and S. Y. Puspitasari, "Analysis Of The Implementation Of Mvvm Architecture Pattern On Performance Of Ios Mobile-Based Applications," *JIPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform.*, vol. 8, no. 1, pp. 59–65, Feb. 2023, doi: 10.29100/jipi.v8i1.3293.

[13] M. N. H. A. Jaelani and Y. Asriningtias, "Optimalisasi Aplikasi Financial Tracker berbasis Mobile dengan Penerapan Design Pattern MVVM untuk Mengelola Keuangan," *Edumatic J. Pendidik. Inform.*, vol. 8, no. 2, pp. 545–554, Dec. 2024, doi: 10.29408/edumatic.v8i2.27709.

[14] T. Haryanti, N. A. Rakhmawati, and A. P. Subriadi, "Measuring the digital transformation maturity level independently with the design science research methodology," *Syst. Eng.*, vol. 27, no. 1, pp. 159–176, 2024, doi: 10.1002/sys.21714.

[15] P. Antunes, N. H. Thuan, and D. Johnstone, "Nature and purpose of visual artifacts in design science research," *Inf. Syst. E-bus. Manag.*, vol. 20, no. 3, pp. 515–550, Sep. 2022, doi: 10.1007/s10257-022-00559-2.

[16] H. Weigand, P. Johannesson, and B. Andersson, "An artifact ontology for design science research," *Data Knowl. Eng.*, vol. 133, p. 101878, May 2021, doi: 10.1016/j.datak.2021.101878.

[17] A. Iskandar, M. Mansyur, A. Obaid, and S. Meiramova, "Development and user evaluation of an android-based e-learning system in higher education," *Discov. Educ.*, vol. 5, no. 1, p. 443, Apr. 2026, doi: 10.1007/s44217-026-01317-z.

[18] F. Setyatama and I. K. Andrianto, "Rapid Application Development (RAD) Method for Developing Clinical Laboratory Information System (case Study: PT. Populer Sarana Medika)," *JEECS (Journal Electr. Eng. Comput. Sci.*, vol. 3, no. 2, pp. 421–430, Dec. 2018, doi: 10.54732/jeecs.v3i2.129.

[19] R. Stephens, "Normalizing Data," in *Beginning Database Design Solutions*, Wiley, 2023, pp. 163–201. doi: 10.1002/9781394320646.ch7.

[20] R. F. García, "MVVM: Model–View–ViewModel," in *iOS Architecture Patterns*, Berkeley, CA: Apress, 2023, pp. 145–224. doi: 10.1007/978-1-4842-9069-9_4.

[21] A. Intana, K. Tantayakul, and P. Kaewnaka, "BlackBoxTestGen: An Automatic Black-Box Test Case Generation Framework," *Computers*, vol. 15, no. 5, p. 263, Apr. 2026, doi: 10.3390/computers15050263.

[22] W. (Bill) Albert and T. S. (Tom. Tullis, *Measuring the User Experience: Collecting, Analyzing, and Presenting UX Metrics*. Elsevier, 2022. doi: 10.1016/C2018-0-00693-3.

[23] M. Hertzum, "System Usability Scale: A Meta-Analysis of How SUS Relates to Workload, Task Time, and Error Rate," *Int. J. Hum. Comput. Interact.*, pp. 1–14, Feb. 2026, doi: 10.1080/10447318.2026.2625260.

[24] N. W. M. -, P. S. -, and K. A. S. -, "Implementasi Arsitektur Model-View-Viewmodel (MVVM) Dalam Pengembangan Sistem DIL-Miclearn Berbasis Mobile," *J. Inform. dan Tek. Elektro Terap.*, vol. 14, no. 2, Apr. 2026, doi: 10.23960/jitet.v14i2.9161.

[25] R. Alfandi, Y. Darmayunata, R. Hardianto, and M. Arief Hasan, "Implementasi Arsitektur MVVM Dalam Pengembangan Aplikasi Study Club Duck.Sc Menggunakan Metode Rest Api Berbasis Mobile," *SEMASTER Semin. Nas. Teknol. Inf. Ilmu Komput.*, vol. 4, no. 1, 2025, doi: https://doi.org/10.31849/nxz0dy38.

[26] M. A. Subandri, F. P. Putra, and A. Tedyyana, "Analysis of the Provider State Management Pattern for Performance and Maintainability in a Flutter-based Mobile Pharmacy Application," *SISTEMASI*, vol. 15, no. 4, p. 1322, Apr. 2026, doi: 10.32520/stmsi.v15i4.6237.